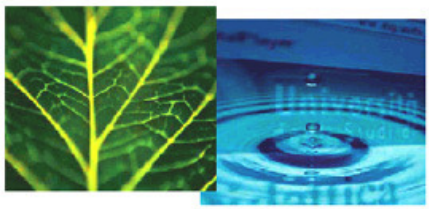
PhD Dissertation



International Doctorate School in Information and Communication Technologies

Department of Information Engineering and Computer Science (DISI) - University of Trento

Extraction and Exploitation of User Goals and Intentions for Querying and Recommendation

Dimitra Papadimitriou

Advisor:

Prof. Velegrakis Yannis
Università degli Studi di Trento

Thesis Committee:

Prof. Fabian M. Suchanek, Telecom ParisTech, France
Prof. Felix Naumann, Hasso-Plattner-Institut fur Softwaresystemtechnik, Germany
Prof. Kostas Stefanidis, University of Tampere, Finland
Prof. John Mylopoulos, University of Trento, Italy
Dr. Georgia Koutrika, HP, USA

Abstract

Users are often found in situations where they need to make selections from very large collections of items. These items may be digital artifacts, e.g., web pages or forum posts, or digital representations of real world objects, e.g., products or people. There is a great deal of techniques for assisting users in making such selections. However, the plethora of systems and the size of the item collections makes the ability to provide the users with the items that really meet their standards in terms of interestingness and usefulness, a challenging task.

We are dealing with the problem of providing items of interest to the users as response to explicit user requests or in the form of recommendations by exploiting a factor that has been poorly investigated so far in information systems: the goals for which items are intended, i.e., the goals for which items have been generated or produced; and the goals that may lead the user to "consume" them, i.e., the goals that s/he is willing to fulfill. The items may not be just items but interactions with items or actions that the user may be interested in performing. In this dissertation, we provide the required background and framework for exploiting goals in building better data managements systems. Within this context, we study three different problems.

First, we are dealing with the problem of finding posts of interest (related posts) given a post-query in forums within user communities. Forum posts consist of segments each one serving a different goal that the author had in mind to communicate to the reader through the text. Therefore, plain content comparisons often fail to retrieve posts of interest, or they retrieve posts that despite the similar content are not related to the post-query. Instead, we have developed a goal-aware matching approach that uses content similarity over intention-based segmentations, i.e., over segments that are intended for different communication goals to perform more effective comparisons.

Second, we are dealing with the goal-aware recommendation problem. This problem, opposed to the post matching mechanism to which we have referred earlier does not consider domain specific characteristics; thus it can be applied to any domain. The goal-aware mechanisms we have developed handle the diverse goals that the user can fulfill by first recognizing the intended user goals, deciding the priorities among them, and by quantifying the benefit of each item.

Last but not least, we are dealing with the problem of building a goal implementation set from texts where users describe how they managed to fulfill certain goals in their real

life. We have applied our technique on textual descriptions from a goal-setting site.

For each solution we have designed, implemented and extensively evaluated models, algorithms and techniques that deal with all the individual tasks that are required for a goal-aware approach: the identification and extraction of goal-related information in the examined data sources, the modeling of the derived information, the matching of the user's request *or* previous activity *to the goal model elements, and finally the exploitation of this matching into the forming of the* system's response*.*

The goal-aware techniques have been found to retrieve items that would not have been considered by the traditional techniques giving to the user a different and more complete view of the item collection. Moreover, the scalability of the techniques and the efficient structures and indexes that we use to store and retrieve the items alongside the goal-related data allows us to meet the requirements of modern online systems.

Keywords[data management, goal-aware systems, retrieval, recommendations, querying, goal models, goal, intention, matching, segmentation, indexing]

Dedicated to those who have always had faith in me.

Acknowledgements

First, I would like to thank my advisor, Prof. Yannis Velegrakis for his guidance and support and for sharing with me his valuable experience all these years. I believe that the lessons I have learnt thanks to our collaboration will be proven precious not only in my next career steps but in my life in general. I would also like to thank Prof. John Mylopoulos for all his support. I am grateful for all the great opportunities they both gave me, and for the possibility to work on problems that I found interesting and challenging, and to meet and discuss with researchers from all over the world. My sincere thanks also goes to Dr. Georgia Koutrika for sharing her valuable experience and knowledge with me, for her encouragement and support to my ideas, and for the opportunity she and Jerry Liu gave me to visit their lab and join their team.

I would also like to thank the rest of my PhD committee: Prof. Fabian M. Suchanek, Prof. Felix Naumann, and Prof. Kostas Stefanidis and everyone that has provided me with feedback and comments helping me to improve this work.

A special thanks goes to all the (former and current) dbTrento group members and co-PhD students with whom we had very interesting discussions and shared important experiences and to all my friends for always being there for me even when they were far away.

Finally, I want to express my gratitude to my family, to my dear parents that have always been on my side and raised me to follow my dreams without forgetting the real values of life; to my beloved sister who has been always a source of inspiration and strength as well of joy and optimism; and last but not least to my partner who has been tirelessly supportive and encouraging, keeping me safe and happy. I deeply appreciate their selfishless love and support. Their faith in me makes me a better person.

Dimitra Papadimitriou

Work partially supported by the ERC grant Lucretius 267856.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

A *shoe* is similar to a *hammer* if one is looking around for something *to bang with*, but not if one wants *to extract nails*

Trewin 2000

People are daily facing situations in which they have to make choices from really large collections of items. Selecting the best answer to a search engine query among those satisfying the query conditions, selecting a movie to watch, an item to purchase, or the friends activities to read about in social media, are only some of the most characteristic examples. In order to help people (users under the context of a system) to deal with all these situations, a plethora of systems and services that search and recommend different types of items and information have been developed.

Despite the remarkable progress in recommendation and retrieval tasks the last years, in many cases, users still fail to deal effectively with the information overload. As a consequence a large volume of items remains unexplored and unexploited. Moreover, together with the scientific progress in the related fields, the user expectations have evolved as well. Users, being bombarded with all sort of recommendations, are not pleased anymore with just relevant and accurate information. Therefore, in order to achieve the user approval and satisfaction, there is a need of introducing novel ideas and practices in item retrieval and recommendation systems.

1.1 Problem

Recognizing the open challenges in this area, this thesis deals with the problem of the *searching and recommendation of items of interest* to the user. Items of interest can be also characterized as *related* or *useful*. The problem that needs to be solved here is: considering a collection of items, to find those that are most likely to be of interest to a

user given a number of items that the user has already found to be of interest. The already selected items of interest constitute either the *user history*, or a specific *user request*. We should highlight that the term *item* here is not implying only a physical object, but an *action*, or an *interaction* with a physical object as well. Examples of items are *wear tie* p_j, *read post* d_i. When there is only one type of interaction with an object, e.g., always *read* or *wear*, we refer to the items using only their type. In the previous example, the type is *documents* and *clothes* respectively. On the other hand, when there are more than one types of interactions. e.g., *buy book* b_k, or *read book* b_k, we refer to them simply as *actions* instead of items. Note that actions may be even more abstract, i.e., they may not involve a specific item, e.g., *eat healthy*, *exercise* etc.

1.2 Motivation

In order for a technique of finding items of interest to be successful, it should be in accordance with the policies that humans, herein users, employ in order to select one or more items, i.e., with the *selection policies*. Existing approaches mainly embrace two policies. The first one advocates that users select items being aware or seeking certain characteristics of the items; while the second advocates that user selections are determined by the past selections of other users.

In the first category, we find traditional IR and Web searching techniques that employ features such as weighted terms (e.g., based on the boolean vector space model or tf-idf weighting scheme) [Robertson et al., 1998], or upper-level features such as hidden topics (e.g., latent Dirichlet allocation) [Blei, 2012; Zhou et al., 2011], links, html tags and so forth ; as well as content-based recommendations that employ domain-specific characteristics, (such as “material” , or “color” for clothes, or “author” for books), and/or other auxiliary data [Adomavicius and Tuzhilin, 2008], [Balakrishnan, 2010], [Fouss et al., 2007].

On the other hand, examples of techniques that adopt the second policy are those exploiting groups of items appearing together, e.g., frequent patterns and association rules [Sandvig et al., 2007], or similarity to the selections of the user's neighborhood [Balakrishnan, 2010], [Deshpande and Karypis, 2004], [Li et al., 2004], [Rendle et al., 2010]. Alternatively, other methods that are embracing this policy monitor the interactions of users with the items and infer how likely a user would be interested in a certain item based on models that capture previous user selections [Sadikov et al., 2010], [Patterson et al., 2003].

The approaches that adopt one of the two policies make decisions that are determined by the preferences of the users. For instance, a user that has read a post that talks about the new release of Ubuntu may be willing to read a post with similar content, and a user that has shown similar preferences with a subset of users that have all read posts about

Ubuntu may also like to read it as well. Likewise, a user that has read almost all the books of an author is likely to read the latest book of the same or a similar author.

Similarity based on previous user preferences indeed affects user selections. However, it can deprive users of a great number of interesting and useful items. For instance, the action of reading a book about the reinforcement of the self-esteem will not be recommended to a user that has read a book about mediterranean cuisine, and another about nutrition. The reason is that those books neither belong in popular subsets, nor have been selected (preferred) by the current user's neighborhood, nor share common characteristics with the books to which the user has shown her/his preferrence, i.e., those that s/he has already read. However, reading a book about self-esteem in combination with the other two books can be of great help to a user that, for instance, is willing to lose weight or to adopt healthy eating habits.

In some cases, the use of plain similarity may even lead to items that are neither interesting nor useful. Considering the previous example, a user that has read a book about self-esteem in her/his effort to lose weight is not necessarily interested in other psychology books as well. Even more often plain similarity can be misleading in cases of items that consist of text. Although similar words or topics in, for instance, two text-items do indicate a connection between those items, the purposes that those words or topics are used for can change their meaning radically.

The above examples indicate that humans both produce (or generate) and consume items to serve certain purposes, or else goals. In fact, based on studies in psychology and social sciences, goals have been found to be the factor that can *rationalize and provide context and meaning* to all human actions [Austin and Vancouver, 1996], [Newell, 1982], [Sadri, 2014], [Sen et al., 1986], [Thompson and McEwen, 1958]. Consequently, neither item selections are random and disjoint events. They are driven by the goals that the items are serving. Therefore, techniques that are retrieving or recommending items of interest to the users can significantly benefit from embracing goals in their selection mechanism, i.e., by adopting a *goal-aware mechanism*.

1.3 Our position

Our position is that items are intended for a number of goals that users take into consideration when they are interacting with them. Therefore, by going beyond the traditional modeling and analysis of data and by introducing to the problem of finding items of interest this novel selection policy that is based on *the goals that are (or can be potentially be) served by those items*, recommendation and retrieval methods can bring into the surface items that the user may not have considered or could not have discovered otherwise.

Our position has been reinforced by the fact that goals have already been exploited in different systems and applications in computer science [Papadimitriou et al., 2016]. However, existing works are focusing on inferring the current goal of the user so as to handle her/his next actions, e.g., the system may automatically perform the inferred action [Armentano and Amandi, 2009], or to facilitate the action in any way, e.g., by providing useful information [Maragoudakis et al., 2007], or even to prevent users from performing an action [Schank, 1995]. For instance, in web (keyword) query answering techniques, the life cycle of a goal (e.g., to be informed or to watch a video) starts with the user request and ends when the user stops searching because s/he is satisfied or quits [Jansen et al., 2008; Kofler et al., 2016].

On the contrary, we consider that every item serves a number of goals alone or together with other items independently from the current user. We have moved the focus from "what the user will do next" to "what each item or action is intended for". We see goals as an integral part of the data that not only changes the way that items are seen and represented but should be embraced in the core functionality of the used algorithms. Of course we do not neglect the user. The user activities or the user requests constitute the input of the goal-aware system. Specifically, the system, considering that the user is willing to fulfill a number of goals (without necessarily taking into consideration or knowing the available items and/or goals), and that the items are serving a number of goals matches those two accordingly and generates a list of items of interest for the user.

1.4 Research Challenges and Contributions

Following a goal-aware approach of solving the problem of finding items of interest introduces a number of demanding challenges that regard the identification and extraction of goal-related information, the modeling of the goal-related information, and the development of the algorithms that capture and exploit this information effectively and efficiently.

Regarding the extraction of goal-related data, we distinguish two directions that can be followed to deal with these issues. In the first one, the required information to build the model is being mined directly from the item collection together with other characteristics (ref. 3.6), while in the second case the information is derived from another data source. There is a great variety of data sources that are available in structured or unstructured format (ref. Section 4.3). For instance, web sources in text format with possibly some structures (e.g., html tags) can be proven very rich sources of information (ref. Section 2.2.5). We have elaborated solutions that fall in both cases.

Next the contributions of this thesis are briefly summarized.

Goal aware systems.

Goals have been used in different fields in computer science with each field approaching the topic from its own perspective and having different objectives. Intelligent interfaces [Lieberman, 2009] is one such example, in which the system tries to guess what the user intends to do from a single or from a number of user actions, and then adjusts the interface options accordingly so that the user can achieve the intended task faster and more conveniently. Story generators and computer games [Gold, 2010; Meehan, 1981] are extensively using techniques to guess the user goal and adjust the way the story is going to evolve in the future. Predictions of interactions and next actions, such as queries, likes, purchases, downloads and comments [Chelmis and Prasanna, 2012; Cheung and Lee, 2010; Sadikov et al., 2010] are also benefiting from the consideration of goals. Goals have also been used in real environment applications, for instance, in applications that help the elderly. In order to exploit goals in building better data managements systems, one should first be provided with the required background under a single prism that takes into consideration the special needs of systems meant for storing, querying and retrieving information.

In this dissertation, we study the different goal-related approaches together under such a prism, without focusing only on a specific area or discipline. We provide a complete and global study alongside a generic formal definition of goals and the related concepts. We intend to enable the exploitation of goal management methods in fields where goals have not considered in the past, significantly improving that way their functionality. We show that this is possible and present the mechanisms for achieving it.

Finding related posts to a post-query. Users often generate text documents known as posts where they share with other users all kinds of information and describe their personal stories but they also describe their problems, or their experience and knowledge on different issues. As a result they are also found in situations where they need to explore very large collection of posts. Browsing and keyword-searching are the most common options to do so. However, having detected a post of interest, this post can be used to retrieve more items of interest related to the initial post (post query). This list can cover several aspects of the initial post in a similar way, while other aspects in a complete different way offering to the user a richer, more complete picture of the item collection. This need becomes more prevalent in *forums* within online user communities. The reason is that an effective solution to the problem of finding related forum posts given a post-query enables the exploration and exploitation of the previous experiences of millions of users all over the world in a great range of different domains. Such a solution requires a

retrieval and ranking mechanism that will measure the relatedness of the post-query to the rest of the posts in the collection and will match it to the most related ones in an efficient way.

In this dissertation, we are dealing with the current problem following a different approach from traditional approaches that perform content comparisons across the post contents as a whole. We see each post as a set of segments, each written intended to express a different communication goal. We advocate that the relatedness between two posts should be based on the similarity of their respective segments that have the same intention. This means that it is possible for the same terms to weigh differently in the relatedness score depending on the intention of the segment in which they are found. We have developed a complete goal-aware intention-based matching technique that in its first step segments the posts by monitoring a number of text features and identifying the parts in which significant jumps occur indicating a point where a segmentation should take place. For the segmentation procedure we have introduced a number of feature groups (referred to as communication means), measures for evaluating the borders, and three different border selection mechanisms. Subsequently, the generated segments of all the posts are clustered to form the intentions and then similarities *across the posts* are calculated through similarities *across segments with the same intention*. For the clustering, any well-known technique can be employed, but it should be used a post representation that captures the distribution of the features that constitute the communication means features. For the post representation, we have introduced 2 types of weights resulting in a vector representation of 28 dimensions that effectively captures the required information. On the other hand, the similarity is estimated using a variation of a well-known weighting scheme the BM-25 that we have adequately modified to consider the intentions of the segments where each term belongs. For the ranking we have developed an algorithm that first generates ranking lists for each intention and then combines the results into a single list. The time efficiency of the ranking is ensured by a number of indexes that are built on the terms of the segments. We experimentally illustrated the effectiveness and the efficiency of our segmentation method and of our overall matching approach.

Recommendations through goal exploitation. The aim of a recommender system is to estimate the utility of a set of items belonging to a given domain, starting from the information available about users and items. Recommenders, based on the past preferences and/or other context information [Balakrishnan, 2010; Deshpande and Karypis, 2004; Li et al., 2004; Rendle et al., 2010; Koutrika et al., 2009; Sarwar et al., 2001; Stefanidis et al., 2012], suggest items from popular or similar item subsets, or items strictly similar to those of the past. On the other hand, recommenders that consider goals should focus

on the identification of the items that are more appropriate based on the goals that the user will be willing to accomplish. Adding goals into the equation can modify the whole perception of what a recommender system is.

In this dissertation, we are formally introducing the dimension of goals into the item recommendation problem. We do not consider items alone but interactions with items or more abstract actions that lead to the fulfillment of goals. As a first step, we generate a space of candidate goals inferred from the actions performed by a user. The candidate goals define indirectly a space of candidate actions for recommendation. We rank the candidate actions to form the recommendation list for the user activity. The ranking requires a solution that deals with the fact that actions can be combined with different action sets to accomplish more than one goal, and also the fact that a goal may be fulfilled through different sets of actions. We have devised three different strategies on ranking the candidate actions, namely the *Best Match*, the *Focus* and the *Breadth*, depending on the preference on the way the goals should be fulfilled. To model the associations among actions and goals through goal implementations (i.e., sets of actions that lead to the fulfillment of goals) we introduce a goal model, referred to as association-based model. The association-based model considers each action set that leads to the fulfillment of a goal as a hyper-edge that connects the actions within the set and at the same time with the goal(s) those actions serve. In practice, we implement our model using a number of indexes. We have performed a comprehensive experimentation where our technique has been found to retrieve novel yet useful items.

Building goal implementation sets. *How-to* articles and *personal success stories* on the Web constitute a valuable source of information about goal implementations, i.e., sets of actions that are needed in order to accomplish certain goals. Recent years have witnessed great successes in information extraction from text and the organization of the extracted knowledge. However, information extraction techniques have so far focused mainly on the creation of ontologies through entity identification, event detection and relationship extraction. Similarly works that regard goal related data also aim to transform the data into a structured form, like an ontology [Jung et al., 2010],[Pareti et al., 2014],[Ryu et al., 2010], a taxonomy [Chulef et al., 2001],[Strohmaier et al., 2009],[Smith and Lieberman, 2010a] or an activity model (ref. Section 2.2.5). Therefore, the problem of extraction of sets of goal implementations can not be handled through the existing approaches.

In this dissertation, we suggest effective and efficient methods for identifying the text expressions that correspond to actions, in a way that copes with the different alternatives that an action may be described in the text without taking into consideration any structural information. We apply our *goal implementation extraction* method on a dataset

collected from an online platform where users describe their success stories, i.e., how they managed to accomplish their goals, and we evaluate it. The results confirm the effectiveness of our approach.

1.5 Application domains

Goals can provide a formal foundation for going beyond the traditional modeling and analysis of data, and can help build better systems and services for the end user for different application domains. Below we briefly present those that constitute the focus of interest of this research.

Retrieval and Data Mining Systems. User posts can cover any aspect of the daily life or any specific topic and interest such as technology, science, psychology. Especially posts in forums span a great range of domains like health (*Medhelp*), traveling (*TripAdvisor*), law (*ExpertLaw*) and technology (*HP support forum* or *IBM*). Therefore, the functionality of *finding related posts* can bring great benefit to users being in completely different situations. Posts are usually organized in threads of a large number of posts. One or more of these posts may contain the solutions to the issue expressed in the initial post. The solution-posts may be either marked as correct by the users, or automatically classified by the system [Jenders et al., 2016]. This way, users can also seek solutions and make decisions regarding diverse problems by exploiting other users' experience. For instance, a user reading a post on a technical problem in a customer care site could find related forum posts that describe similar situations and alternative solutions. Or, someone with a health problem reading a medical forum post where a user is describing similar symptoms could find additional related forum posts that contain different opinions, explanations, and various courses of actions. This functionality also offers businesses the ability to connect and support their customer base.

Recommender systems. Goal-aware recommenders can offer a more surprising and richer experience for the user since the associations of items due to goals are not obvious nor known by all users, especially when goal models capture diverse implementations for the same goal. Goal-aware recommenders can be employed in any application domain as long as items (or actions) are organized into sets that operationalize (i.e., lead to the fulfillment of) one or more goals, forming goal implementations. For instance, an online food market can discover items that can help a user to adequately exploit the items s/he has already bought in order to fulfill one or more goals. Under this context, the goals that are served by the food-items can be food recipes or nutrition goals. Other examples of application domains where sets of items operationalize certain goals are: sets of different clothing items that constitute outfits, or books, that independently from the categories

where they belong are known to serve real-life goals, e.g., a book about mediterranean cuisine, another about obesity, and a third one about the reinforcement of the self-esteem can support the goal of losing weight (ref. Section 1.2), at the same time the first book about mediterranean cuisine together with a guide for wine selection and the benefits of olive oil can support a user' s goal to adopt mediterranean food habits. We further discuss such options in Section 4.3. An interesting type of goal implementations that opens up the road for novel recommendation services is that of social data that describe how users fulfill real-life goals such as to learn english, to visit Athens, to eat healthy and so forth (ref. Section 2.2.5).

Social Data and Services. Beyond goal-aware recommenders, mechanisms that extract information related to goals (information regarding actions, goals, and how goal are fulfilled, i.e., goal implementations) can be proven valuable for the analysis of data from social networking sites and applications. For instance, for analyzing the content that a specific user shares online to extract her/his goals. The detection of information about the user goals and actions in social media can trigger several marketing services. Users with similar goals could get connected to discuss common problems and get motivated. Moreover, special services can be implemented for groups of users with common goals.

1.6 Outline

Chapter 2 presents our study of the techniques of inferring and exploiting goal related information that have been used in different fields in computer science, and explains what is required for introducing goals in a new unseen scenario. The next two chapters, namely Chapters 3, 4, introduce, formally define and provide complete goal-aware solutions, and extensive evaluation to the problems of finding related forum posts and item (action) recommendation, and Chapter 5 presents a technique for text extraction that is used to build goal implementation sets. We summarize our findings and provide insight for the open challenges in Chapter 6.

1.7 Scientific Outcome

- D. Papadimitriou, Y. Velegrakis, G. Koutrika, and J. Mylopoulos. Goals in social media, information retrieval and intelligent agents. In IEEE 31st International Conference on Data Engineering (ICDE) 2015 [Papadimitriou et al., 2015] (tutorial).

- D. Papadimitriou, G. Koutrika, J. Mylopoulos, and Y. Velegrakis, The Goal Behind the Action: Toward Goal-Aware Systems and Applications. ACM Transactions on

Database Systems (TODS) 41(4), 23. 2016 [Papadimitriou et al., 2016](survey).

- D. Papadimitriou. Goal-aware data management for retrieval and recommendations. In 32nd ICDE PhD Workshops. IEEE Computer Society, 2016 [Papadimitriou, 2016].

- G. Koutrika, D. Papadimitriou, and S. J. Simske. Matching of an input document to documents in a document collection. U.S. Patent No. 20,160,299,891. 13 Oct. 2016 [Koutrika et al., 2013](patent).

- D. Papadimitriou, G. Koutrika, Y. Velegrakis, and J. Mylopoulos Finding Related Forum Posts through Content Similarity over Intention-based Segmentation IEEE Transactions on Knowledge and Data Engineering (TKDE) (accepted for publication), 2017.

- D. Papadimitriou, G. Koutrika, and Y. Velegrakis, Recommendations through Goal Exploitation (under submission).

- D. Papadimitriou, G. Martella, Y. Velegrakis, and G. Koutrika. Goal implementation extraction (under submission).

Chapter 2

Goal-aware Systems

This Chapter, provides an extensive and comprehensive study of the techniques and the ways that goals have been used so far. Section 2.2 provides the list of different techniques that have been used so far for goal modeling and recognition and also explains how this information is actually exploited. Subsequently Section 2.3 describes a series of applications in which goals have or can be used, and the benefits that they can offer to the functionality of these applications. It also provides some examples of how the tasks and concepts described in the unified framework work in a real application scenario. Prior to all these, in Section 2.1, it provides a unified overview of goals and the related concepts offering a formal framework that can work as a reference point to the concepts and challenging tasks in any goal-oriented approach and system. All the methodologies and application scenarios are discussed under the prism of this framework.

The organization of the sections of this chapter is based on the different tasks that a *goal-aware system* should perform. The main different tasks are three. First, a goal-aware system has to be able to model and store user goals. Second, it should be capable of identifying the goals given a set of observed actions that the user has performed. Recall that goals are rarely mentioned explicitly or communicated, but instead exist in the users' mind. Last, but not least, the system should be able to exploit the recognized goals and adapt its functionality and output accordingly.

Goal modeling. To exploit goals, a system needs, first of all, to obtain a collection of goals alongside the knowledge of how these goals can be fulfilled. Goal modeling is challenging. Several approaches have been studied in applications and environments phenomenally disconnected. For example, we meet goals in: (a) software for computers or other electronic devices, e.g., for providing intelligent interfaces; (b) the Web as a collection of resources, where goals can be incorporated into query answering; (c) limited physical environments, where one or more sensor-based intelligent systems act, e.g.

activity recognition sensors that send personalized activity reminders; and (d) physical locations monitored by sensors, e.g., an airport monitored for suspicious behavior.

We categorize goal modeling approaches according to the source of data used for the construction of the model. In particular, we consider those that are based on:

– *Complete records* of all the required information about goals, actions and plans,
– *Taxonomy records* containing actions matched to classes of a goal taxonomy,
– *Corpuses* containing all the information for a *subset* of the goals, actions etc.,
– *Behavior theories* providing all the required information about human goals and actions within a specific environment.

Goal recognition. To recognize goals, one needs to observe user actions within an environment and identify patterns that lead to the satisfaction of user goals. User actions include queries, purchases, menu item selections, free-text input, preference statements, publishing multimedia objects, moves in a natural environment or moves of certain human parts, user interactions, and so forth. Data mining techniques, such as association rules that could capture knowledge in the form of "a set of actions X is followed by the set of actions Y with high probability", can provide useful correlations among observed actions and system conditions. However, they cannot answer the question of "what the user wants to achieve with these actions?", i.e., identify and assign a goal to these actions [Wilcox and Bush, 1992]. Goal representation is tightly coupled to goal recognition, thus, we study them together.

Goal exploitation. Understanding a user's goal as the user interact with the system can help the system to adapt its operation, personalize its responses, and facilitate the user in achieving their goal. For instance, imagine a user of a text editor, who opens the submenu for tables but changes nothing, then checks the printing settings a couple of times. The user is probably trying to print a table so that it fits the page. Knowing the user goal, the system can facilitate, e.g., by highlighting related menu options. Exploitation of goals can save the user from performing irrelevant steps, or putting extra effort, and the system from unnecessary operations and costly computations [Armentano and Amandi, 2009; Carberry, 1983; Gold, 2010; Zhe et al., 2010]. Or it can make available knowledge, information or any other type of response that is derived consider the current goal of the user [Broder, 2002; Maragoudakis et al., 2007; Sadikov et al., 2010].

The exploitation of goals in practice is strongly dependent on the application scenario. We categorize existing approaches across two dimensions:

– *Exploitation Through Dynamic Environment Changes*, i.e., changes in the environment states by taking into consideration the inferred goal(s).

– *Exploitation Through System Responses*, i.e., responses to user requests performed

by algorithms that embrace goals into their core functionality (i.e., the system performs its tasks considering the inferred goal).

Goals can be exploited at any given point of a system's lifecycle. During requirement specification, goal-oriented approaches aim at capturing the objectives a system should achieve [Mylopoulos et al., 1999]. Since the focus of this study is on systems, we do not consider such works. Furthermore, we do not consider studies on goals and human behaviour from a psychological and sociological perspective. There also exist a few surveys on goal exploitation, and they focus mainly on goal recognition. One of these surveys studies ways through which a goal can be recognized by observing the actions that a system agent performs and provides an overview of techniques on how the recognized goals can be used in decision making [Anh and Pereira, 2013]. Along the same lines, another survey emphasizes on plan recognition and probabilistic methods [Armentano and Amandi, 2007], while a third one approaches the challenge of goal recognition through logic-based formalisms [Sadri, 2014]. All the above works are agent-oriented, thus, they consider approaches typically employed by agents. They have an Artificial Intelligence (AI) flavor, ignoring issues like performance or usability. Furthermore, they have not considered works in areas like recommendation systems or information retrieval, and is hard to see how the presented approaches can be adopted by other areas. There also exists a survey focusing on the special needs of multimedia searching [Kofler et al., 2016] that covers certain aspects of text retrieval as well. However, we are approaching these techniques considering a generic formal definition of what a goal is and all the related concepts.

2.1 Key Concepts

Before studying the way goals have been used, it is necessary to establish a common terminology and formally define a number of concepts.

We assume the existence of a countable set $\mathcal{U}$ of *actors*, that can be persons or (software) agents. Actors live and operate in a environment, performing *actions* that affect it. *Environments* can be natural, such as a room monitored by sensors, or virtual, i.e., created by a computer program. To realize the effects that actions have on an environment, we assume that an environment has a countable number of states. The states are specified by a number of factors, that are modeled as variables. In particular, we assume the existence of a countable set $\mathcal{V}$ of variables. Each variable $v \in \mathcal{V}$ is associated with a domain D_v, the values of which are the possible instantiations of the variable and are referred to as the *states of the variable*.

Definition 1 *An* environment *E is a finite set $\{v_1, v_2, \ldots, v_k\} \subseteq \mathcal{V}$. The variables $v_1, v_2, \ldots, v_k$ are referred to as the* environment variables *and the number k, denoted as $|E|$, is the* cardinality *of the environment. A* state*, or* instance*, of the environment is a set $\{S_{v_1}, S_{v_2}, \ldots, S_{v_k}\}$, with $S_{v_i} \in D_{v_i}$, for i=1..k.*

The symbol $\mathcal{S}^E$, or simply $\mathcal{S}$ will denote the set of all possible states that an environment E can be. The state of an environment is changed by actions performed by the actors or by external factors. For example, a variable that represents time changes independently of any actor actions.

Definition 2 *An* action *is a function act:$\mathcal{S}^E \rightarrow \mathcal{S}^E$, expressed as a conjunction of "v=S_v" pairs, where v is an environment variable of E and $S_v \in D_v$.*

For brevity, we will write $(v_1, \ldots, v_i) \overset{act}{\rightarrow} (S_{v_1}, \ldots, S_{v_i})$ to denote $v_1 = S_{v_1} \wedge \ldots \wedge v_i = S_{v_i}$. Furthermore, if for an action act, it holds that $act(S)=S$, where $S \in \mathcal{S}^E$, then the action will be said to have no effect on the environment state. Finally, the symbol $\mathcal{A}$ will be used to denote the set of all possible actions.

Example 1 *Consider an environment E=\{place, time, status\} that describes the position of a person at some point in time, and whether she is alone or not. Assume that at the current moment it is night and the person is at the sea alone. The current state of the environment is S_{curr}=\{"Sea", "Night", "Alone"\}. An action $(time) \overset{act_1}{\rightarrow} (\text{"}Day\text{"})$ will bring the environment into the new state S_{new}=\{"Sea","Day", "Alone"\}, while the action $(place) \overset{act_2}{\rightarrow} (\text{"}Mountain\text{"})$ will lead into the new state S_{new}=\{"Mountain", "Night", "Alone"\}*

By definition, an action can always be executed. However, there are many practical scenarios, in which it is important to restrict when this can happen. For this reason, an action may be associated to a set of *preconditions* for its execution.

The actions that the actors perform are not random but are performed for a reason, i.e., the actor wants to achieve a *goal*.

The term goal has been defined in different contexts as the point that marks the end of a process, the purpose towards which an endeavor is directed, an objective[1], or a desired state of affairs. All these definitions converge into a generic description of a goal as one or more desired states described through some common properties.

Definition 3 *A* goal *g in an environment E is a boolean expression of environment variables of E. The goal is* fulfilled *(or* achieved *or* satisfied*) in a state S of the environment*

[1] http://dictionary.cambridge.org, http://oxforddictionaries.com

E, and denoted as $S \models g$, if after the replacement of each environment variable in the boolean expression g with its state in S, the expression evaluates to true. The set of all possible goals is denoted by $\mathcal{G}$.

Example 2 *Consider the environment of Example 1 and the desire of the person to go to the mountains during the day. This desire can be modeled as the goal g: $((place = \text{“}Mountain\text{”}) \wedge (time = \text{“}Day\text{”}))$. Note that the goal is independent of the state of the variable* “status”.

In order to fulfill their goals, actors are typically making plans on what actions to perform. This is known in the literature as the *operationalization* of a goal [Dalpiaz et al., 2014].

Definition 4 *A* plan *is a sequence $\langle act_1, act_2, \ldots, act_n \rangle$ of actions. The* operationalization *of a goal g in an environment E with a state S is a plan $\langle act_1, act_2, \ldots, act_n \rangle$ for which $S' = act_n(act_{n-1}(\ldots act_2(act_1(S))\ldots))$ and $S' \models g$. Such a plan is also referred to as a* successful plan *for this goal. Lack of a successful plan makes the goal* infeasible.

Example 3 *The plan that consists of the two actions mentioned in Example 1, in any sequence, is a successful plan for the goal g: $((place = \text{“}Mountain\text{”}) \wedge (time = \text{“}Day\text{”}))$ since the final state of the system after the execution of these two actions is $S' = \{\text{“}Mountain\text{”}, \text{“}Day\text{”}, \text{“}Alone\text{”}\}$, which satisfies the goal. The plan with the additional action $(status) \xrightarrow{act_3} (\text{“}WithCompany\text{”})$ is also a successful plan since it brings the system in the state $S'' = \{\text{“}Mountain\text{”}, \text{“}Day\text{”}, \text{“}WithCompany\text{”}\}$, which also satisfies the goal.*

The *implementation* (or execution) of the plan is the execution of its actions. The *life-time* of a goal consists of the states that the environment goes through from the time a goal was set to the time a state was reached in which the goal is satisfied.

People often talk about how close they are in achieving a goal, or to what degree a goal is fulfilled. Hence, there is a notion of proximity between the current state of the environment and the set of states that satisfy the goal, i.e., the set $\mathcal{S}_g = \{S |\ S \models g\}$. To quantify this proximity, we assume the existence of a *fulfillment function*, $scr_g : \mathcal{S} \rightarrow [0, 1]$, which is a scoring function with $scr_g(S) = 1$, for every $S \in \mathcal{S}_g$, and $scr_g(S) \neq 1$ otherwise. Such a scoring function is typically goal- and application-specific, since it depends on what factors of the environment are considered important and how much. A goal is *partially fulfilled* with respect to a state S, if the value of this goal fulfillment function for the specific state is in $(0, 1)$.

In some cases, actors set goals that have no clear specification on when they are fulfilled. This type of goals is called *soft goals*, a term originally proposed in the field

of goal-oriented requirements engineering [Mylopoulos et al., 1999]), to be distinguished from the "hard" goals introduced in Definition 3, which are based on a boolean function.

Definition 5 *A* soft goal *is a function* $g{:}\mathcal{S}^E \rightarrow \mathbb{R}$.

Intuitively, a soft goal provides a way to quantify whether one state of an environment is preferable to another, but there is no state in which it can be said that the goal has been satisfied. Due to this fact, a fulfillment function cannot be computed. However, between two states S_1 and S_2, it is typically said that the state S_1 has better fulfilled the soft goal g than S_2 if and only if $g(S_1) > g(S_2)$.

Example 4 *An example of a soft goal is web searching in which users search for resources in order to get informed about a topic. For this purpose, the actions they perform are: to submit keyword queries, and click on the available web resources. It is hard to say that at some point the goal has been fulfilled (i.e., that the user knows everything about the topic). However, in an environment defined by features such as the diversity of the web resources, the position of the keywords within a page etc. (ref. Section 2.2.2, especially Model Construction & Table 2.1 for more examples), it can be defined a function having as parameters a subset of the environment variables, i.e., the goal function, to return whether one environment state is preferable to another. The environment states are actually matched to a number of web resources. According to the selected state, the respective web resources become available to the user.*

Setting goals means typically some commitment to perform a sequence of actions for achieving that goal. The term *intention* is used to capture that commitment.

2.1.1 System Components & Tasks

We consider systems that allow us to store, retrieve, and discover information and knowledge from a data repository. In the physical or virtual workspace defined by the system, actors perform actions, such as submitting searches or sending requests (e.g., for services), and the system responds by changes in the environment, services or data in its environment. The *actors* are users or other systems/applications that interact with the system. Thus, the systems we consider are *interactive*.

The pink-colored part of Figure 2.1 illustrates the typical components of such a system. The bottom part is the data repository in which all the data is stored. This data is accessed by the main component of the system, the *System Engine*. The System Engine implements the main functionality of the system, i.e., runs the main algorithms. A data analytics system would include in that component algorithms for data mining, text analytics and

statistical analysis that will be running over the stored data. A recommendation system would comprise different recommendation methods, while a traditional database system would entail methods for accessing the underlying data based on the specifications of the actor's queries.

Actors' actions support a goal that the actor wants to achieve. For example, a user that poses a query "europabank, credit card, pay" is likely looking to pay the monthly statement balance rather than finding the cost for a new card. Respectively, the user would expect results that are different from the results when they try to find the cost for a new card.

A system that ignores goals, i.e., a *goal-agnostic* system, misses the big picture, i.e., "why is the actor doing this?", hence it cannot help the user as effectively as possible. In our earlier web search example, that would signify an increased user effort to achieve their goal. Knowing actors' goals can help the system understand their actions and adapt its behavior and functionality to a goal faster and more effectively, resulting in better system resource usage as well as user experience. A *goal-aware* system would require the components to record the goal-related information, analyze it, and then use the analysis results to recognize the goals of the actors, and respond accordingly. These components are shown in green in Figure 2.1.

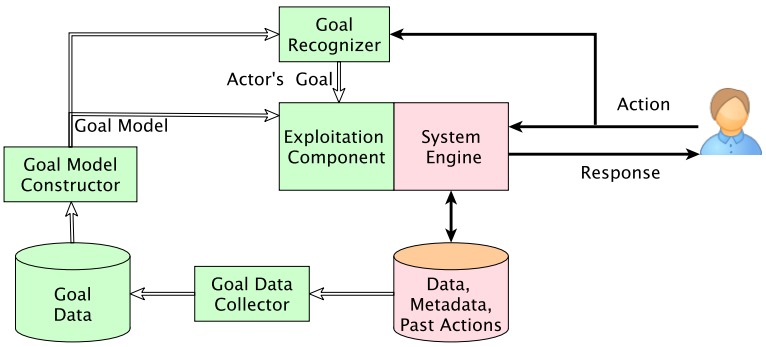


Figure 2.1: A goal-aware System Architecture.

Goal Data Collector. In any data management system that allows interactions, it is common to track past user actions in the *data repository*. In a similar way, a goal-aware system also keeps information on the goal each such action serves (whenever it is available) and when these goals have been fulfilled. Consequently, the *data repository* is extended with the *goal repository* that contains information about the goals and the ways they can be operationalized, their actions, preconditions and effects. Actions may be recorded directly in log files or indirectly, i.e, via changes in the values of environment variables (e.g., temperature tracked by a sensor). Alternatively, goal data can be gathered from

experts or user annotations. Figure 2.2(a) shows several alternatives for collecting goal data.

Goal Model Constructor. Goal data can be used to create the so-called *goal model*, which is a model used to recognize and subsequently to exploit the actor goals. The goal model construction can be done either in a top-down fashion, where experts or actors themselves explicitly state their goals and actions, or in a bottom-up fashion, where models are constructed by *observing* and analyzing the actions of the actors in the system. The goal model is constructed offline and may be updated periodically as new actions and their goals are recorded into the goal repository. Figure 2.2(b) shows alternative approaches for building goal models. The goal data collector and the goal model constructor comprise the *goal modeling* operations of a goal-aware system.

Goal Recognizer. With the goal model constructed, the next challenging task is the *goal inference* or *recognition*.i.e., the ability to infer the goal(s) that an actor is currently pursuing by observing her actions [Sadri, 2012]. The task is challenging because the actions provide only partial information. The idea is to recognize the goal way before all the actions that operationalize the goal have been completed. Depending on whether or not the actor wants to disclose the goals, the task can be characterized as *intended*, in which the actor tries to communicate her goals to the system, *keyhole*, in which the actor is unobtrusively observed and does not attempt to impact the recognition process, and *adversarial* in which the actor is hostile and tries to hide her actual goal [Geib and Goldman, 2001]. An example of the intended case are the natural language dialogues, where speakers explicitly try to communicate their goals. An adversarial example is the cases of computer security and information warfare, where the actor tries to hide her intentions and the system tries to predict them in order to identify possible attacks. Finally, an example of the keyhole case is in query answering systems where users interact normally with the system without explicitly expressing their goals nor hiding them from the system. Moreover, apart from the user's effort to keep her/his goals unrevealed, there are cases where we have *partial observability* of the environment for reasons such as sensor limitations, uncertain action logs, or privacy issues. In these cases, the matching between the actions that are actually performed and what is observed in the environment is not deterministic; making goal recognition more challenging [Hoelzl et al., 2012; Keren et al., 2016a]. For instance, an action may be matched to more than one sets of effects on the environment. Figure 2.2(c) shows alternative goal recognition approaches for different goal models.

Plan recognition is an extension of goal recognition that aims at identifying not just the goal but also the plan followed by the observed actor in order to achieve her goal. In the basic case it is assumed that an actor is pursuing a single goal using a deterministic

set of actions, hence, a plan can be identified by matching these actions against the actions in a goal model such as a plan library representing the operationalization of the various goals. This decision cannot be done with a complete certainty, either because the observations match partially and are not enough to make a firm decision, or because the representations themselves are incomplete or uncertain.

Goal Exploitation Component. The recognized goal (and possible plan) can be exploited at run-time by the system algorithms. The module can select the responses to provide in order to drive the actor towards the fulfilment of the set goal. We highlight that the system "responses" are not necessarily returned data. They may be actions that the system takes even if not explicitly requested by the actor, for example as performed by intelligent interfaces. This means that the goal exploitation module may provide different responses to the same request if the identified goals are different or if different plans are used for the operationalization of the goals. In a query answering system (e.g., a Web Search Engine), the results retrieved with the existing techniques are all related to the query in general, but based on the goal that has been recognized some may be more important than the others. Thus, the results can be further processed to keep only those that will enable or facilitate the goal fulfillment. To achieve this functionality, the main system component (system engine) that performs data selection should be aware of the identified goal, its operationalization choices (i.e., the plans that lead to goal fulfillment), and the interaction history of the specific actor.

Note that goal exploitation is not necessarily an additional processing step. Existing algorithmic techniques can be adapted (or novel techniques can be designed) to take goals into consideration by performing the respective reasoning (that requires goal modeling and recognition) that has just been described.

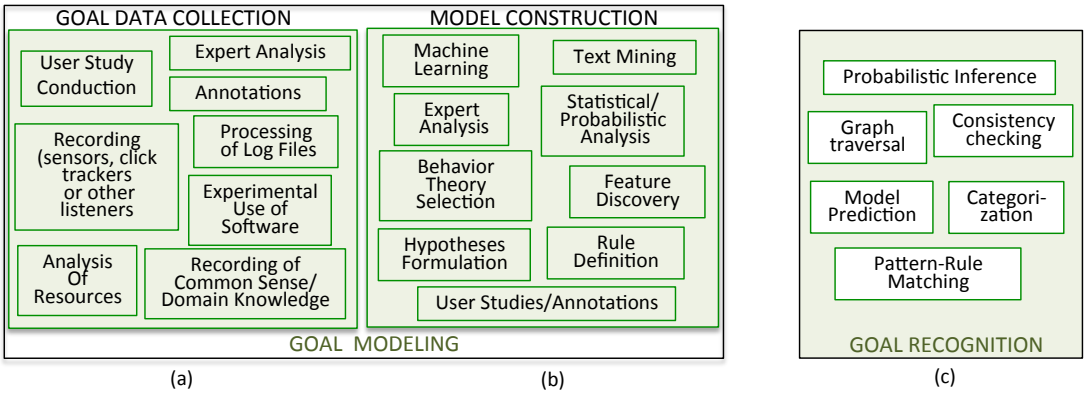


Figure 2.2: Methods that may be employed during Goal Modeling and Goal Recognition.

2.2 Goal Modeling and Recognition

Goal models, and by extension goal recognition techniques, are not restricted to a certain application scenario. The same modeling method can be used in different goal exploitation scenarios. The available data based on which the goal models are constructed together with the needs of the system determine the goal modeling approaches that can/should be used. Based on the *main source of goal data* used for the construction of the models, goal models can be clustered into models derived from: *complete records*, *taxonomies*, *corpuses* (training data), and *behavioral theories*. In these methods, different ways to collect the goal data are applied as we will see (Figure 2.2(a)). We also consider another alternative, collecting goal data from *text corpuses*. Text analysis is not offering a complete solution for goal modeling and inference. However, as we will see, challenging issues in terms of data management get raised by the discovery of goal knowledge in text data.

For each goal model type, we present the most common techniques for constructing the model based on the goal data, and for inferring the goals based on the current observations (goal inference).

2.2.1 Based on Complete Records

Approaches *based on complete expert records* assume that *experts* provide all the information needed for building a model sufficient to match any set of observations to a latent goal (and possibly to a plan) within the examined environment. Goal recognition proceeds as follows. Initially, all the goals that require the actions observed so far are considered as candidates. As the actor continues to perform more actions, some of the candidate goals become logically infeasible due to missing actions in their implementation plans, due to violated preconditions or due to other observations. These goals are excluded from the candidate set reducing the search space. During the check for the goal infeasibility, logic-based reasoning (mainly logic abduction) can be employed to provide explanations of the observations [Sadri, 2012] when this is possible. The goal recognition task does not always lead to some conclusion but when it does, it returns one and only one goal. We identify three categories of approaches based on complete records: (a) plan libraries, (b) consistency graphs, and (c) action-centric representations.

Plan libraries

Plan libraries could be characterized as a set of recipes describing alternative plans for implementing a set of goals in which the developers of the goal recognition system are interested. They contain information about (i) the set of actions that an actor is allowed

Problem Domain	Hacking
Goal notation	(theft), (vandalism)
Actions	**Description**
$(reconnaissance) \overset{recon}{\rightarrow} (\text{“}true\text{”})$	make a reconnaissance, scan the system to determine vulnerabilities
$(breakedIn) \overset{break-in}{\rightarrow} (\text{“}true\text{”})$	exploit the system weaknesses
$(root) \overset{gain-root}{\rightarrow} (\text{“}true\text{”})$	gain entry break in escalate privileges gain root
$(exportDataRoot) \overset{steal}{\rightarrow} (\text{“}true\text{”})$	export desired data root
$(exportData) \overset{mod-webpage}{\rightarrow} (\text{“}true\text{”})$	export desired data
$(deleted-logs) \overset{clean}{\rightarrow} (\text{“}true\text{”})$	hide traces of presence

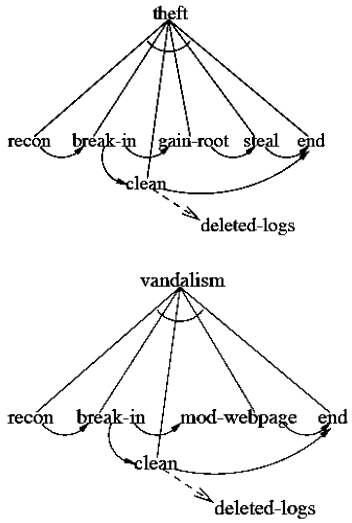


Figure 2.3: Example of Hierarchical Plan Library.

to perform and (*ii*) the preconditions of each action alongside its effect on the environment [Carberry, 2001a]. Plan libraries have to be *complete* i.e., to exhaustively describe all the actions that may be performed in the domain under study and all the alternative plans for all the possible goals. Moreover, they have to be *correct* since they lack mechanisms for handling inconsistencies [Sadri, 2012].

Model Construction. The construction of a plan library demands a lot of effort by experts who master the problem domain. In general, experts try to organize the goals and actions into plans in a way that enables the generalization of the plans to cover new facts. Even psychological theories about how human observers understand the actions of others have been used in the task [Schmidt et al., 1978]. In many cases, the domain experts perform closed-world reasoning [Kautz, 1991], i.e., they isolate a part of the world in which they are mainly interested and focus on the minimum sets of independent plans that explain the observations and are sufficient for fulfilling the goals of the observed agents.

Simple plan libraries may be represented as rules of the form $g \Rightarrow act_1, act_2, \ldots act_n$, where g denotes a goal, and $act_1 \ldots act_n$ is the sequence of actions which comprise the operationalization of the goal. Actions in the literature may be represented as predicates, e.g., *land*(*jet101*, *airbase1*) describes the action of *landing* of *jet101* to *airbase1* [Sadri, 2012]. Respectively, in a simple environment E (ref. Section 2.1), the action $land \in \mathcal{A}$ for instance would be defined as: $(jet101_{Loc}) \overset{land}{\rightarrow} (\text{“}airbase1\text{”})$, where $jet101Loc \in E$ and $D_{jet101Loc}$={airbase1, airbase2, airbase3}.

Plan libraries can take a hierarchical form as well. Figure 2.3 depicts a simple hierarchical plan library capturing hacker goals in a web system [Geib and Goldman, 2001]. On

the left, the goals and the actions (together with a short description) are shown. On the right, the actual library is illustrated as two diagrams that correspond to two plans for the goals: (*theft*) and (*vandalism*). As we see, the actions in the plans are not sequential; there exist links that determine *ordering constraints* among the actions.

Moreover, dashed lines represent the fact that a change in the environment is observed after the execution of an action (i.e., the action postconditions are illustrated). For instance, if *clean* is executed, a change in the environment will be observed, i.e., *deleted-logs* value will change from "false" to "true". This information about action effects can be critical to inferring the execution of unobserved actions (all actions except from clean are not observed). For example, in Figure 2.3, the ordering constraints allow us to conclude that in order for *clean* to be performed without being observed, an earlier unobserved *break in* should also occur.

Goal Inference. Goal inference using plan libraries requires the detection of the plan that is consistent with the current observations. The correctness and completeness of the library is of major importance for the inference task. If the observations can be matched to more than one plan, the system should either wait until one or more actions exclude all the candidate plans but one, or return all the matching candidate plans. This approach is extremely sensitive to noise. One misidentified action may cause the exclusion of the real plan from the candidate set of solutions. However, inference in plan libraries is not always straightforward. In cases of adversarial recognition, in which hostile agents try to hide their actions from the system, there is no fully observable sequence of actions. Thus, probability distributions are introduced in the inference technique by algorithms such as Poole's PHA to first infer unobserved actions, based on the observed actions or the state of the system, and then infer the most probable goal [Geib and Goldman, 2001]. The search space can be limited by considering the ordering of constraints and/or by excluding disabled actions. An action is considered to be disabled when in all the plans of the library, it is preceded by actions that have not been observed.

Consistency graphs

Consistency graphs are graphs that consist of (*i*) proposition nodes that store the values of the environment variables, (*ii*) nodes representing actions, (*iii*) nodes representing goals, and (*iv*) edges representing possible connections between nodes. Instead of constructing a complete plan library that includes all possible plans related to every possible goal, to build a consistency graph one should *focus on defining what constitutes a valid plan.* In other words, how the allowable actions can be combined to a plan that fulfills a goal according to the structure (i.e., the environment), the restrictions (i.e., action preconditions and

postconditions), and the system functionality (i.e., the allowed actions and goals).

In contrast to goal recognition systems with complete libraries, where a goal is *consistent*, if there is a plan that starts with actions already observed leading to the goal, consistency graphs are able to recognize new plans as well [Hong, 2000]. However, since the action restrictions are not modeled, consistency graphs cannot capture causal links among actions and goals. That makes them more appropriate for explaining past actions rather than making predictions.

Model Construction. Initially, all the actions and goals that are feasible in the examined environment are recorded by the domain experts and are inserted as nodes in the consistency graph. Then, all the action nodes and the goal nodes are fully connected to each other without checking for inconsistencies, i.e., without checking whether the sequences of actions (plans) that are connected to a goal violate any conditions and if they actually lead to the fulfillment of the graph. Inconsistent goals are then repeatedly pruned from the consistency graph. The consistency control may be performed by the experts or automatically.

Goal Inference. The main idea is to reduce the set of candidate goals by eliminating the goals that cannot be explained by the actions that the actor performs. If more than one goal can be explained by the observed actions, consistency graphs cannot return a result since only one consistent explanation is possible.

Action-centric representations

Action-centric representations, originally proposed for classical planning problems, have recently been for goal and plan recognition by exploiting the progress in modern plan synthesis [Sun and Yin, 2007], [Ramırez and Geffner, 2009].

Model Construction. In contrast to the construction of plan libraries, where domain experts record all actions possible, in action-centric representations, the modeled actions are the outcome of all the possible combinations of the environment variables to a set of pre or post conditions. Thus, the size of the state space is exponential to the size of the set of environment variables. However, can be selected and stored only the actions that have an important impact on the environment. The modeling is done using propositional logic. Environment variables are modeled as propositions, the states of the environment as a set of propositions connected with logical symbols such as AND ($\vee$) and OR ($\wedge$), and action effects are modeled as two sets: the first set indicates which propositions will be removed from the environment state after the action is performed, and the second which propositions will be added. There are also models that instead of propositions use first-order literals.

Initial State

(and(garbage)
(clean hands)
(quiet)
(not present)
(not dinner))

Goals

g1	(and(dinner)(present)(not garbage) (quiet))
g2	(and(dinner)(present)(garbage)(quiet))
g3	(and(dinner)(not present)(garbage)(quiet))

Actions

Preconditions	Effects	Short Explanation
(clean hands)	(dinner)	COOK. Before: your hands should be clean. After: dinner is ready.
(not present)	(present)	WRAP UP. Before: present is not ready After: present is ready.
(garbage)	(and (not garbage) (not cleanHands) (quiet))	CARRY GARBAGE. Before: garbage exists. After: there is no garbage, the agent's hands are unclean and there is silence

Figure 2.4: An example of a state-variable representation of a simple goal recognition problem.

An example is the STRIPS models that are expressed in the homonymous modeling language. STRIPS language has been initially suggested to represent planning problems for a specific software, a planner called STanford Research Institute Problem Solver (STRIPS) [Fikes and Nilsson, 1971], but since then it has been used as a tool for representing the environment in planning and goal recognition problems independently of the STRIPS planner. In STRIPS models, often it is preferred to store only the action postconditions, i.e., the changes that occur in the environment, instead of storing the complete outcoming environment states for reasons of efficiency. Moreover, except from states and goals, STRIPS includes operators. Operators represent the combination of two or more actions that cause a state transition that is considered important for the system.

Figure 2.4 shows a state-variable representation of a problem examined by Sun et al. [Sun and Yin, 2007] in which an actor may potentially prepare dinner, throw away the garbage and wrap up a present for his girlfriend. The representation consists of 3 actions, 3 goals and the initial state. The actor goals are combinations of the propositions: (*dinner*) i.e., dinner is prepared, (*present*), i.e., present is wrapped up, (*garbage*) i.e., garbage is not thrown away and (*quiet*), i.e., the agent's girlfriend is not waken up. The goal (*and*(*dinner*)(*present*)(*not garbage*)(*quiet*)), for example, describes that the actor wants to clean the room and prepare both the dinner and the present without waking up his

girlfriend.

According to Section 2.1, the environment in the above problem can be defined as: E={*dinner*, *present*, *garbage*, *quiet*, *clean hands*}. The domains of each of the respective environment variables $v_i \in E$ are D_{v_i}={"true","false"}. Moreover, the goals g1, g2, g3 are determined by the respective environment states, i.e., in g1 (dinner) will be translated into *dinner*="true". Thus, g1 will consist of two environment states of $\mathcal{S}^E$: g1={{"true", "true", "false", "true", "true"}, {"true", "true", "false", "true", false}}. As the state of variable *clean hands* (which is the last variable) is not explicitly stated in g1, both environment states are desired.

The construction of the goal model starts with the initial state of the environment. After having produced all the possible actions (derived actions), the transition graph is constructed layer by layer, with the first layer being the initial state. Specifically, all the derived actions are examined and if the preconditions of an action are satisfied or there are no other inconsistencies, the state is updated according to the effect of the current action. Inconsistencies include: actions with inconsistent effects (effect-effect), actions where an action effect interferes with the precondition of another action (effect-precondition), or actions. The process is run recursively until the planning graph stabilizes.

Goal Inference. To infer goals from the derived transition graphs, search space algorithms are used, such as breadth-first search (BFS) and A^*, in combination with heuristics to boost up performance. According to the BFS strategy, search is performed level by level; first, all the existing sibling nodes (nodes of the same level) are visited, then the next level of nodes is examined, and the procedure goes on until a node that represents a goal consistent with the observations is visited. A^* strategy reaches the node representing the consistent goal by following the path of the minimum cost according to a cost function, such as minimizing the length of the path which contains all or some of the action nodes that have been observed. The starting search point is the current state of the system. Ramirez and Geffner showed how algorithms originally designed for planning can be slightly modified and used for plan recognition over a domain theory [Ramırez and Geffner, 2009].

2.2.2 Taxonomy-based

Taxonomy-based approaches require the existence of a taxonomy of the possible goals, i.e., a set of goal categories, within the system. The categorization is performed by experts and requires studying of the existing actions (i.e., those that have already been posed), identifying the actor goals, and then building the taxonomy. In the area of goal-aware query answering, where they have been extensively used, the goal taxonomies

were derived from *extended user studies* using questionnaires and *interactive tools* on web browsers tracking user moves such as clicks and form submissions in combination with *expert knowledge*. Examples of goal taxonomies [Broder, 2002; Kang and Kim, 2003; Lee et al., 2005; Rose and Levinson, 2004] are briefly presented in Section2.2.2.

In taxonomy-based approaches, there exist 2 types of actions: (a) the actions that initially trigger the functionality of the system, and (b) the actions that become available after the system response. The first type of action are the *user requests* or *queries*. For instance, in a web search engine, the requests are the *keyword queries*, while the actions are the clicks on the web resources, e.g., pages, snippets, that the system returns to the user after the query is posed. The actor requests are used for goal inference while the actions performed after the system response may be tracked by the system to evaluate the actor's satisfaction. Ideally, the actor would be satisfied by clicking a single web resource. Thus, the plan would consist of a single action. However, more actions, i.e., clicks on the top related resources returned, may be required, creating longer plans.

Model Construction. Once the categories of the taxonomy are decided, the model that will classify a user query into one of the goal classes should be built. To build the model, experts select a number of environment variables that are considered appropriate for grouping the requests into the classes of the *goal taxonomy*. Manual classification can be used to understand the user goals and whether it is feasible and meaningful to incorporate them in the existing system. To automate (at least partially) this laborious task, the analysis of the involved resources can be employed, e.g., in Web Information Retrieval query logs and snippets have been used. The selected environment variables are then used in rule-based annotators [Jansen et al., 2008; Lee et al., 2005; Li et al., 2006] or to train automatic classifiers, such as Support Vector Machines (SVMs) with RBF (Radial Basis Function) kernel [Baeza-Yates et al., 2006; Herrera et al., 2010], can be built.

The selection of environment variables (features) has to be performed very carefully, since for different domains the accuracy of the classification may increase or decrease significantly depending on the features. This is particularly evident in Web Searching [Herrera et al., 2010]. For Web queries, the variables more widely used fall in five categories: (i) anchor-text based features, (ii) features regarding urls, (iii) query-based features, (iv) features based on user past clicks, and (v) page-content features (ref. Section 2.2.2 and Table 2.1). They are typically extracted directly from the resource collection, or from snippets retrieved via classical retrieval techniques, or query logs. The selected environment variables constitute the main features. However, for each goal class, there may exist additional features that are essential for the definition of the goals to be inferred, e.g., the number of different involved web resources, or their diversity for informational queries for

instance.

Goal Inference. Goals in taxonomy-based approaches are *soft goals*(ref. Section 2.1, Definition 5). Thus, goal inference is about defining a function over the environment variables. Considering the set of environment variables selected in the model construction step, the constructed model (i.e., the rule-based annotator or classifier) matches every new request in the system to one (or possibly more) of the goal categories. The goal class sketches (or else partially defines) the goal since for each goal class, there is a set of conditions on the environment variables. The goal takes its complete form by extra conditions on the environment variables based on the request itself and the goal class. Hence, goal inference requires two tasks: (a) categorization of the request to one of the categories of the taxonomy, and (b) definition of a function that captures conditions on environment variables based on the goal class and the request.

Some works have treated goal inference from user queries as a problem of query reformulation. They define goals in web search as sets of semantic concepts [He, 2010] or as sets of "verb-object" pairs derived from the sentences that are implied by the queries [Chang et al., 2006]. Although these approaches showcase interesting results, the focus of this work is on goals that can (even approximately) describe a desired state of the environment, so we do not consider them further.

[*User satisfaction*]. An important aspect in taxonomies has been the evaluation of goal inference. Even if the model is accurate, it may not correctly classify a *request*, mainly because of the subjective nature of soft goals. In contrast to hard goals (ref. Definition 3), soft goals (ref. Definition 5) are defined by a function $g{:}\mathcal{S}^E{\rightarrow}\mathbb{R}$. To define this function, one needs to know whether the actor will characterize a plan as successful, which is not possible to know in advance. To cope with this challenge, analysis of action patterns (e.g., sequences of queries or clicks within user sessions) have been employed by the information retrieval community. These analyses have been based on Markov Models [Hassan et al., 2010] or on Hierarchical Conditional Random Field techniques [He, 2010].

Examples in Taxonomy-based Approaches

In the area of goal-aware answering, to come up with the goal taxonomy, experts actually studied the information needs (sometimes called user intentions) that drive users to search on the web [Baeza-Yates et al., 2006]. Examples of *information needs* are *to be informed*, *to navigate to a site*, *to execute a transaction*, or *to get advice*. There has been also work on converting their textual descriptions, which are actually the labels of the goal classes, to a set of features for automatically assigning an incoming query to a goal class [Baeza-Yates et al., 2006; Herrera et al., 2010; Jansen et al., 2008; Lee et al., 2005; Li et al., 2006].

The Broder taxonomy [Broder, 2002] was the first created. It was implemented by examining whether it is feasible to identify what users expect from a web search engine so as to consider their search successful and stop submitting similar queries. The outcome of that work was a taxonomy of user queries that reflects a categorization of latent user goals. The taxonomy describes three types of web queries: informational, navigational and transactional. *Informational* queries consist of terms that describe or capture vague notions, e.g., "*bugs*", or consist of specialized terms, e.g., "*Peruvian Dubia cockroach*". Both types of queries indicate that the desired target is a collection of links which will enlighten the user on the subject. *Navigational* queries consist of query terms that describe a specific url, e.g., the query "*american airlines home*" indicates that the user's most probable target is *http://www.aa.com*. Finally, *transactional* queries contain terms that indicate that the target url enables a transaction such as downloading a file, buying an item or watching a video. For instance, with the query "*Athens photo*" the user most probably expects to get direct access to image files related to Athens.

User surveys proved that the consideration of Broder's query taxonomy into the selection of the query results has a positive impact on user satisfaction [Broder, 2002]. Consequently, further research was triggered towards this direction and differentiations of the taxonomy have been developed towards more detailed ones [Jansen et al., 2008; Rose and Levinson, 2004] or more abstract ones [Baeza-Yates et al., 2006]. For instance, Rose and Levinson elaborated Broder's taxonomy by dividing informational queries into five subcategories [Rose and Levinson, 2004]: (*i*) directed queries that express specific questions, (*ii*) undirected queries that aim at retrieving all the available information about a topic, (*iii*) list queries aiming at getting a list of candidates, (*iv*) find-queries aiming at locating real-world services or products, and (*v*) queries aiming at getting advice, ideas, suggestions or instructions. Going back to our earlier example informational queries,"*bugs*" is an undirected query while "*Peruvian Dubia cockroach*" is a directed query. Similarly, transactional queries are divided into four categories that express what exactly the users want to do. In particular, the user may want to (*i*) "download", (*ii*) "view an item such as a video", (*iii*) "interact via another program or service", or (*iv*) "obtain a resource" (video file, text file etc.).

The aforementioned taxonomies were *evaluated by* user studies and the classification of the sample queries has been performed by experts based on information about the queries, on the results returned by commercial search engines, on the user clicks on the result list, as well as on other actions performed by the user before and after the submission of the query [Rose and Levinson, 2004]. The reasons that determined the experts' classification decisions in some cases remain unclear, but there have been efforts to clarify and record them [Jansen et al., 2008].

Env. Variable Types	**Examples of environment variables**
Anchor text [Herrera et al., 2010], [Lee et al., 2005], [Fujii, 2008]	similarity of the query with the top similar anchor texts
Urls of the web collection [Lee et al., 2005]	similarity of the query with the url (important for navigational queries)
Query formulation [Herrera et al., 2010], [Jansen et al., 2008]	num of query terms$\geq$2 $\rightarrow$ informational queries
Past user clicks for the same query [Jansen et al., 2008], [Lee et al., 2005]	skewness of click distribution e.g., for navigational queries: only one click
Cue query terms and "important" terms of the web collection [Herrera et al., 2010], [Jansen et al., 2008]	domain suffixes (e.g."edu"), terms related to: pictures, games etc., to interactions such as buy, chat $\rightarrow$ transactional queries; terms such as "how to", "ways of" and general terms $\rightarrow$ informational queries

Table 2.1: Environment Variables For Query Answering Systems

Based on the goal class of a query (action), there are different answering policies expressed in the goal definition by conditions on the environment variables. For instance, for informational queries, goals should be satisfied by diverse web resources that cover the query from different perspectives, contain complementary and controversial information, and offer various levels of comprehension (for broad, deep or quick understanding). Thus, when a query is matched to this goal class, the inferred goal should be defined as a function on variables such as resource diversity, content diversity, or size of resource. On the other hand, for navigational queries, goals should contain conditions on features describing click streams from logs, since click distributions reveal whether users consider a site to be the "expert" for certain queries. The function that defines the goal can be next used to reorder the results of a web search engine (ref. Section 2.3.1).

2.2.3 Corpus-based

In *corpus-based* methods, the *goal data* contains a set of alternative plans for a set of goals: the *plan corpus*. The plan corpus is used as a training set for statistical models that can make inferences for future observations. There is no ground truth about the

environment; uncertainty expressed in probabilities is the factor that rules the outcome of the recognition process [Russell and Norvig, 2003]. The trust on systems with probabilistic output under difficult critical circumstances is still an open issue [Atkinson and Clark, 2013]. Thus, corpus-based approaches have been criticized when used in real-life domains such as health, defense and transportation and are delegated to difficult and critical for safety tasks, or tasks of high-cost in time or money, or in general of high impact on human lives. Nevertheless, these methods enable goal inference in environments where it is too expensive or infeasible to gather complete and certain knowledge about all the goals and the potential corresponding plans that may be followed by the observed actors. They only require a *plan corpus*, which will constitute a sufficient training data set for developing an efficient statistical model. The two most widely used classes of probabilistic models in these cases are *Markov models* and *Bayesian networks*.

Markov Models

Markov models in their general form consist of nodes representing random (stochastic) variables and edges modeling conditional dependencies among the variables. The values of the random variables may be observed (known values) or may be inferred (unknown values). In the context of goal-aware systems, the values of the *random variables* describe the *environment state*. The main inference task of Markov models used for goal inference is to compute the conditional probability of a sequence of observations given some evidence, i.e., to check to which extent the current observations would be justified, if we assumed that the variable to be inferred had a specific value.

In Markov models, it is assumed that the probability that a random variable (i.e., an environment variable) will have a certain value in the future can be computed by observing only the recent past of a set of observations, i.e., that an observed action act_i is only dependent on the current goal g and the n precedent observations (i.e., observed actions). This assumption is known as the *Markov assumption*. The number of previous observations is called the *order* of the model.

Model Construction. Learning a Markov network requires statistical analysis of the plan corpus to define the following probability distribution functions: (i) the distribution of prior probabilities $P(g)$ indicating the expectancy that a goal $g \in \mathcal{G}$ is being pursued by the observed actor, (ii) the state transition function $P(S_i|S_{i-1}, g)$, where $S_i, S_{i-1} \in \mathcal{S}^E$ that returns the probability that the system will move from the environment state S_{i-1} to S_i given that goal g is pursued, and (iii) the observation function $P(act_j|S_i, g)$, or $P(S_{i+1}|S_i, g)$ that returns the probability that an observation will occur (either act_j will be performed, or environment state $P(S_{i+1}$ will be observed) given that the system is in

state S_i and that goal g is pursued.

Learning the probability distribution functions is typically done by performing a global search in order to figure out *which combinations of environment variables and weights* would give more accurate predictions within the plan corpus [Della Pietra et al., 1997]. This methodology is not efficient and is prone to make only locally optimal choices. To build a consistent and efficient probabilistic model, a two-step methodology has been suggested. First, a model (e.g., decision tree or logistic regression model [Lowd and Davis, 2010; Wainwright and Jordan, 2008]) is built for predicting the value of each variable of the domain with respect to the other variables. Then, the separate models are converted into a single Markov model.

Goal Inference. To infer goals in Markov models, first the goal probabilities are initialized by considering the prior probabilities function $P(g)$. Then every time an observation occurs, the goal probabilities are updated by taking into consideration the conditional probabilities functions defined when the model was created, and the goal g with the maximum probability is selected.

In cases in which the Markov assumption is valid, the probability of goal g can be estimated using the formula (Markov chain rule): $\prod_{i=1}^{n} P(act_i|g)$ (or $\prod_{i=1}^{n} P(S_i|g)$). This rule has the nice feature of composeability: new observations produce conditional probabilities which are simply multiplied with the previous predictions.

In cases in which the Markov assumption is not valid, the complexity of the problem becomes very high (#P-complete) and approximate solutions are required. One widely used method is the Markov chain Monte Carlo (MCMC), such as the Gibbs sampling. MCMC performs probabilistic queries and provides answers by counting the number of samples that satisfy each query over the total number of samples [Wainwright and Jordan, 2008]. By query, it is meant a sequence of observations that is answered given another sequence of observations that is called evidence. The sampling is not performed on the data but is calculated based on the joint probability of the random variables. In contrast to Markov networks that have been learned by methods such as probabilistic decision tree learners (DTSL) [Lowd and Davis, 2010], MCMC allows inference by standard techniques such as loopy belief propagation [Murphy et al., 1999] because their models represent consistent probability distributions.

Markov Model Variations

There are a number of interesting variations of Markov Models that have been used in goal modeling.

N-order Markov Models. In n-order models, the Markov assumption applies for n observations before the current observation while the evidence is the goal g. Hence, to check

which goals explain better the observations, the n last observations are compared with subsequences of observations in the plans of domain goals in the plan corpus. Observations may be either actions or environment states, i.e., $act_1 \ldots act_n \in \mathcal{A}$, or S_1,$S_2 \ldots S_n \in \mathcal{S}^E$. Respectively, plans are either sequences of actions (actions are directly recorded) or of environment states (actions are observed through environment state transitions) from a set of predefined actions or states. The value of n, i.e., the order of the model, should be carefully chosen so as to create an expressive model (larger values of n) and at the same time keep the size of the search space traversable with low cost (smaller values of n). Due to their simplicity, n-order Markov models are very efficient but at the same time they may be ineffective in recognizing goals in cases of complex environments [Blaylock and Allen, 2003].

Variable-order Markov Models. In contrast to the n-order Markov models, in variable-order Markov models (VOMs), the probability of the current goal g is not defined by the same fixed number of previous observations. In other words, the order of the model varies based on the *specific observed realization* in the training data, known as *context*. Therefore, the use of VOM models can increase the accuracy of goal recognition by capturing longer regularities than n-order models, while controlling the size explosion of the search space caused when the n-order is increased. VOMs are learnt over a finite alphabet consisting of all the available actions $\mathcal{A}$. States instead of actions are also possible to be used. Thus, as in n-order models, the plans record either the performance of actions directly or the state transitions before the desired state, i.e., the goal g. Armentano et al. [Armentano and Amandi, 2009] suggested the use of Probabilistic Suffix Trees [Ron et al., 1994] to represent VOMs. For each domain goal, one PST is built to store the subsequences of variable length (plans) that are necessary and sufficient for modeling the corpus plan. Hence, a forest of PSTs is created. In order to optimize space and time efficiency, only the minimal subsequences of observations are preserved. In this case, *goal inference* becomes a classification problem of the sequence of observations to the most probable PSA. However, it is not a common classification problem since early predictions are very important [Armentano and Amandi, 2009].

Hidden Markov Models. Hidden Markov Models (HMMs) are adequate for problems in which the current state of the system is not visible or cannot be identified with certainty, as in computer games and activity detection systems. This is because they do not require complete knowledge of the state of the system. Some or all of the environment variables of the partially observable or hidden states may be estimated based on probability distribution functions over a set of observed variables. These functions are called *output or emission probabilities*. Briefly, to define an HMM, the following probabilities have to be specified: (i) the initial probabilities that a state S_i, where $S_i \in \mathcal{S}^E$ may occur in the first

place, (*ii*) the probabilities that the system may transit from one state S to another S_{i+1} (transition matrix), and (*iii*) the output probabilities.

HMMs may be used as *Hierarchical Activity Models* for activity recognition. First, an ontology of high-level composed activities is built, i.e. the goals $\mathcal{G}$. One of the composed high-level activities g is chosen to be the recognition purpose of the system. Then all low level primitive actions (where primitive actions refer to actions from the action set $\mathcal{A}$ in Section 2.1) that are related to the activity are recorded and are organized in different ensembles, i.e., groups so as to fulfill the recognition goal of the system in different ways. Finally, the HMM is formed with the high-level activity modeled as the hidden state, (i.e., the goal g) and all the related primitive actions as possible emissions of this state according to their probability [Hoelzl et al., 2012].

For example, Figure 2.5 illustrates the Hierarchical Activity Models for two activities: "Coffee Making" and "Table Cleaning" in a physical environment monitored by sensors. To recognize "Coffee Making", 3 sensors are involved: *back*, *right upper arm* and *left upper arm*. Each sensor is monitoring the respective human body part and has been associated with a number of actions. For instance, sensor *back* is selected for inferring action "Walking" based on a measure, called degree of fulfillment, that reflects whether the system trusts the respective sensor to infer the action. The value of the sensor is 0.92 for "Walking" and 0.89 for "Standing". After actions are inferred, the main activity, i.e., the goal g, can be inferred next according to the HMM that is constructed based on the plan corpus. Note that "Table Cleaning" can be detected by the same sensors with "Coffee Making", though associated with other actions: "Clean up", "Interaction with fridge", "Walking" and "Standing".

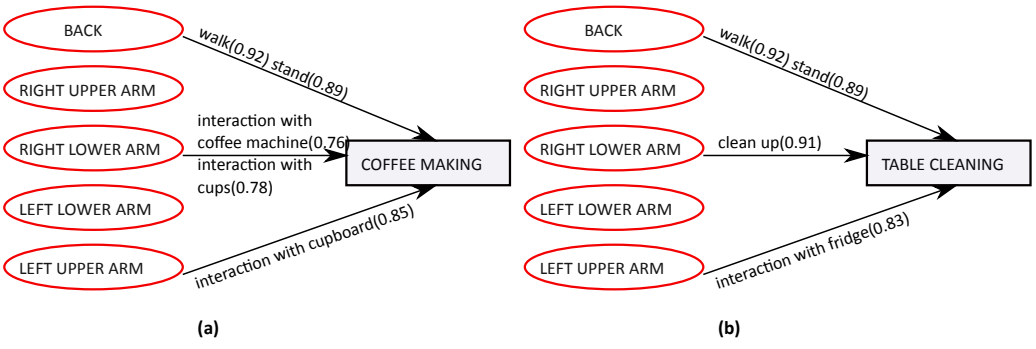


Figure 2.5: Ensembles for goals: (a) Coffee Making, and (b) Table Cleaning.

The use of HMMs requires deep understanding of the problem domain and usually requires very large training samples [Singer and Warmuth, 1996].

Input Output Hidden Markov Models. A variation of HMMs considers additional context information, e.g., the previous satisfied goal. This additional information modifies the

state transition function and the observation probabilities when it is considered necessary. The Input Output Hidden Markov Models, or IOHMMs for short, manage to capture causalities in a similar way Bayesian networks do, and are capable of updating their hidden states in a similar way HMMs do. IOHMMs are good models to be used in domains where there is abundant context information that can be exploited e.g., in computer games [Gold, 2010].

Context information can be taken into consideration also in plain HMMs, by increasing the number of observation categories. However, this choice increases the training time and in addition causes a conceptual mixing of the known variables i.e., the variables showing whether a previous goal has or has not been already achieved with the hidden variable of the model i.e., the current goal.

Markov Logic Networks (MLNs). These Markov networks use Markov Logic (ML), a statistical-relational language that extends finite first-order logic (FOL) to a probabilistic setting. Specifically, they use a set of pairs (F_i, w_i), where F_i is a FOL formula, and $w_i \in R$ is a weight reflecting the significance of the constraint expressed by w_i, to calculate the conditional dependencies between pairs of nodes. The joint probability function is represented as the product of the potential functions. In the context of goal recognition, the MLNs represent the ambiguous causality between actions and goals in the dataset [Ha et al., 2012; Kautz, 1991; Mott et al., 2006]. Kautz [Kautz, 1991] was the first to introduce a formal theory of plan recognition in the context of Markov logic by suggesting a representation that may be transformed to an MLN by adding a binary node (binary variable) for each predicate and by considering the ground logic formulae as features that will determine the transitions into the network.

An example of logic formulae representing constraints in an interactive narrative system in a computer game environment [Ha et al., 2012] is the following set:

1. $\forall t, a : action(t, a) \Rightarrow |\exists g : goal(t, g)| = 1$
2. $\forall t, a : action(t, a) \Rightarrow goal(t, g)$
3. $\forall t, a, s, g : action(t, a) \wedge state(t, g) \Rightarrow goal(t, g)$
4. $\forall t, a, g : action(t - 1, a) \Rightarrow goal(t, g)$

The implicated parameters are: (i) the player (i.e., actor) actions, such as moving to a particular location or opening a door, (ii) the narrative states that represent the player's progress in solving the narrative scenario and (iii) the player's locations in the virtual environment. A narrative state is encoded as a vector of four environment variables, each one representing a milestone event within the narrative. Constraints in ML are divided into hard and soft. Hard constraints have to be always satisfied while soft may be violated.

The first formula represents a hard constraint, which defines that for each action at each time step t, the player has to pursue a single goal g. The second formula represents the prior probability distribution of the domain goals while the rest of the formulae represent the goal g at time step t based on the values of the three parameters: time step t, action type a, and narrative state s. Each formula is assigned a weight that has been learned automatically from the plan corpus using a technique called Cutting Plane Inference [Riedel, 2012]. CPI limits the complexity of large-scale problems by focusing on a subset of constraints.

Bayesian Networks

Bayesian networks (BNs) have been widely used for goal recognition tasks because they manage to capture causality among actions and goals [Horvitz et al., 1998; Huber and Simpson, 2003]. A BN is a directed acyclic graph in which nodes represent the constituent variables of the problem domain and edges the causal relationships or conditional dependencies between pairs of nodes i.e., between BN variables. The entire network can be understood as a representation of the joint probability distributions of all the random variables of its nodes. In a goal-aware system, the constituent variables may be observable or latent environment variables (unknown parameters or hypotheses), or certain actions, or goals. The variables may represent observable quantities, latent variables, unknown parameters or hypotheses. The strength of the connections between the variables is encoded in conditional probability tables. Independent variables are not connected.

For instance, consider a simple narrative virtual environment E=$\{p_1, \ldots p_m, l\}$, where each p_i ($1 \leq$ i $\leq$ m) represents an element that determines the plot of the story, and l represents the location of the user in the environment. Then, the BN variables could be: variables representing sets of the plot elements (narrative states), the variable l indicating the location (e.g., D_l=$\{$"locA", "locB", "locC"$\}$), and a variable m that represents the user moves, i.e., the user actions $\mathcal{A}$, within the virtual environment. Moreover, the links would capture the dependencies among the variables, e.g., a move that results in the change of the narrative state [Mott et al., 2006]. Such a model in the context of an adventure game for instance, can capture goals that involve different locations and plot elements such as being in location "locC", with the plot element p_1: *interaction with the story investigator* being true and the mystery considered as solved (plot element p_k is *solved*). To succeed that, the actor may have performed several moves/actions such as collecting evidence, and exploring different locations.

Model Construction. For building the model, i.e., learning the network, the conditional probabilities can be learnt from real data (training corpus). The structure of the network

can also be learnt to some extent [Heckerman, 1996]. However, most of the time, the structure is manually specified by domain experts. In case of an evolving dataset (continuous introduction of new evidences), the probabilities at each node may be recomputed by propagating the evidence through the edges.

Goal Inference. Bayesian networks can compute the conditional probabilities of the random variables of interest given a set of variables with known values, called evidences. Probabilistic inference using belief networks is NP-hard. To simplify the procedure, it is assumed that the variables are ordered so as for the independency probability assumption to be valid, i.e., for a variable to be conditionally independent of its non-descendants variables given its parents. In this case, as in Markov networks, the Markov chain rule can be applied.

Inference may be efficiently performed using filtering to reduce the number of variables that is taken into consideration such as Rao-Blackwellised particle filtering. The latter is a combination of exact and stochastic inference i.e., sampling is used to reduce complexity but some of the variables (that are considered of greater importance) are excluded from the sampling procedure for higher accuracy [Doucet et al., 2000].

Bayesian Network Extensions

An extension of the Bayesian Network that also includes a temporal dimension is the Dynamic Bayesian Network (DBNs). A DBN [Yin et al., 2008] is actually a sequence of Bayesian networks, each one modeling the dependencies among the variables in a specific time slice. Except from the causal links among the BN variables, there are also intra-slice connections that represent temporal dependencies in consecutive time slices (ref. Figure 2.6). A plan in a DBN is a sequence of actions starting from an action node *act* with an incoming edge from a goal node g. In other words, the links from a node g to a node *act* point out a sequence of actions that implement the goal g.

By using DBNs, more complex models of sequential data, which are hopefully closer to reality, can be represented, and learned. The price to be paid is the increased algorithmic and computational complexity. Parameters must remain the same across time slices so as to model sequences of unbounded length. The simplest way to do exact inference in a DBN is to divide the DBN into slices and then apply some inference algorithm to the resulting static Bayes net.

Figure 2.6 illustrates an example of a Dynamic Bayesian Network. The two dashed squares frame two hypothetical Bayesian networks (one for each time slot) that will occur supposing that the dynamic network is unrolled. The dashed lines represent the connections between two consecutive time slots t-1 and t.

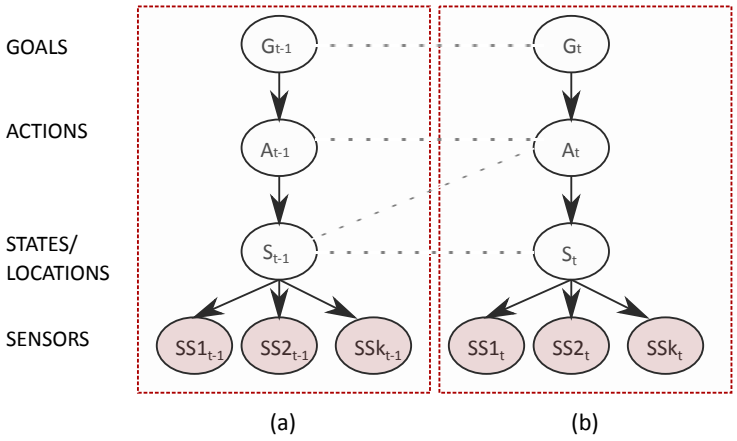


Figure 2.6: An example of a Dynamic Bayesian Network.

Works that use GPS data or data collected from sensors usually have different levels of inference. The lowest level corresponds to "raw" sensor data. In a DBN, the first level models the transitions at intersections and changes of modes of transportation (states), while the transitions at higher levels represent meaningful movements from one location to another (actions) [Patterson et al., 2003]. Moreover, a number of actions leading to certain locations constitute the plans towards the fulfillment of the corresponding goals.

2.2.4 Behavioral theories

In environments, such as those defined in social networking applications where users through their actions and interactions exhibit behaviors similar to those in the real world, the environment states can be determined by a number of environment variables $\{v_1,\ldots v_k\}$: the *motivations* or motivational factors (where $\{v_1,\ldots v_k\}=\mathcal{V}$). Intuitively, a motivation is a factor that drives someone in performing an action. Motivations pre-exist in the environment and they rule consciously or subconsciously user actions and inductively user goals.

Motivational factors and their interdependencies have been defined in theoretical *behavior models* by sociologists [Ajzen, 1991; Fishbein and Ajzen, 1975]. Models that reflect human behavior have been also developed independently by computer scientists [Chelmis and Prasanna, 2012; De Choudhury et al., 2007; Perugini and Bagozzi, 2001].

Two behavioral theories have been mostly used in computer science, the Theory of Reasoned Action (TRA) [Fishbein and Ajzen, 1975] and the Theory of Planned Behavior (TPB) [Ajzen, 1991]. TRA determines two main motivational factors (*i*) the *attitude* of a person *towards a behavior* i.e., her beliefs towards this behavior, and (*ii*) the *subjective norm* i.e., the opinions of the persons that are important to the person under study (who

will approve or disapprove the person in case she follows this behavior). TPB extends TRA by considering also whether the important persons to the user consider this behavior easy and trivial or important and worthwhile; this motivational factor is called *perceived behavioral control*.

Since the above mentioned theories are abstract and general, they can only be used to sketch the initial draft of the model, which is then enriched by motivational factors/variables that researchers consider important for a specific problem. In other words, these theories constitute the framework within which researchers express their hypotheses.

For example, Figure 2.7 illustrates a model for estimating the intention of a user to share knowledge within a business social network suggesting a number of motivational factors. In the graph representation, the nodes represent the environment variables while their dependencies are determined by the stated hypotheses (i.e., each edge corresponds to a hypothesis). For instance, with collective/shared goals, organizational members tend to believe that other employee's self-interest will not affect them adversely and they all contribute their knowledge to help achieve their mutual goals.

Model construction. To build a *theoretical model* for explaining user behavior, and by extension predicting the *user intention* to act towards a goal, the following steps are typically followed: (*i*) selection or formulation of a behavioral theory, (*ii*) formulation of a set of assumptions (hypotheses) for each one of the factors that determine human behavior according to the selected theory; these hypotheses are the variables that define the suggested theoretical model, (*iii*) conduction of a survey on real users to test these hypotheses, and (*iv*) performance of statistical analysis to check the validity and reliability of the model.

The involved variables have to be determined based on the selected behavior model and then a set of causal assumptions or hypotheses that determine the dependencies among the variables are made. *One of the variables expresses always the user intention* regarding a certain behavior i.e., *the behavioral intention*.

The value estimation of some of the variables can be done *by regression relations*. These variables are called *dependent variables* while the variables for which it is impossible to predict their values by other variables are called *exogenous*. For instance, in Figure 2.7, the extensiveness of the social network, the existence of social trust and shared goals are exogenous, i.e., they do not depend on other variables but they impact the dependent variables. The impact one variable has on another is determined by the stated hypotheses that form the behavior model.

Moreover, the in-degree of the nodes in the graph representation reveals whether a variable is exogenous or dependent. Exogenous variables are nodes with in-degree 0 because they are not pointed by other variables/nodes.

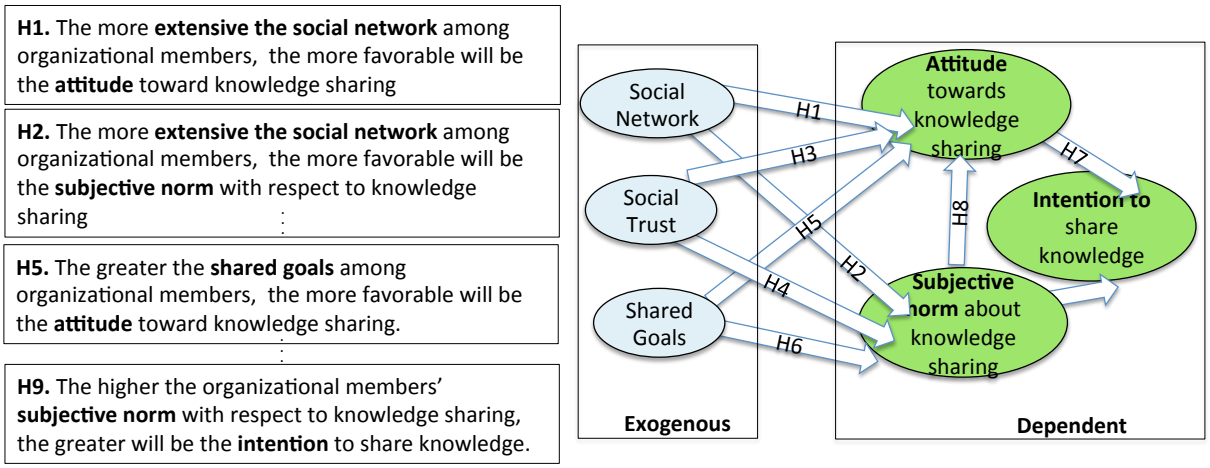


Figure 2.7: Behavioral model for Intention Prediction.

[User survey.] The stated hypotheses, which correspond to the variables of the model, are transformed into one or more questions usually of multiple choice that are answered by a representative sample of users. The results from the user study constitute the ground truth for the analysis. The performance of a user study is inherently problematic for very large datasets, such as data from Facebook (more than one billion of users), though. Consequently, it is feasible only for interest groups of users, such as university students or teenagers in a geographic region [Hsu and Lin, 2008]. By offering free online services, such as simple game applications, companies manage to gather a significant amount of answers by questionnaires that either are part of a game or are required in order to use the service.

[Model accuracy.] In some cases, the consistency of the undirected model is controlled first, i.e., the correctness of the selected variables/factors (confirmatory factoring analysis) [Chow and Chan, 2008]. Then, the accuracy of the whole model is controlled including its dependencies (structural analysis) [Hsu and Lin, 2008].

Goal Inference. The model is used to evaluate the commitment of the actor towards the behavior and the underlying goal. This is done by estimating the value of the variable *behavioral intention* based on the regression relations of the model. The known variables are the motivational factors that constitute the environment state (i.e., the input), while the latent variable is the behavioral intention (i.e., the output). Larger values of behavioral intention indicate more committed actors.

2.2.5 Based on Text Corpus

Text analysis does not offer a complete solution for goal modeling and recognition. However, information about goal achievement is available in multiple sources online with user generated content, such as social networking sites, forums, blogs and online guides. This large amount of data can be systematically analyzed to offer a wealth of information related to various aspects of goal achievement.

Thus, text analysis *can be used for gathering data regarding goals and goal achievement*, a task that is usually accomplished, as previously seen in this section, either directly by experts, or through user studies, experimentation software, and annotation tasks. This way, experts' workload is reduced. Furthermore, since human effort cannot be compared in terms of time efficiency and cost to automatic text mining and NLP methods, the volume of gathered goal data can be significantly larger. *The data volume in combination with the diversity and the special characteristics of goal data makes goal modeling significantly challenging.*

Specifically, instead of (or in combination with) using expert knowledge, aspects such as goals [Castellanos et al., 2012; Smith and Lieberman, 2010b], motivations [Strohmaier et al., 2009], intentions [Castellanos et al., 2012], and actions [Strohmaier et al., 2009] *can be extracted from text.* Furthermore, social data can be mined to discover "recipes" of successful or failed implementations of various goals, to detect user sentiments during the life cycle of a goal and to investigate the impact goals have on social interactions. E-how and wikihow have been used as information sources, e.g., [Pareti et al., 2014],[Jung et al., 2010]. Other datasets have been created by posing queries such as "in order to+*goal description*" to web engines [Strohmaier et al., 2009]. They may also exploit phrases that are known to describe actions to extract connections among goals and actions, or among actions [Smith and Lieberman, 2010b]. Moreover, hand-crafted rules, standard expressions and syntactical patterns (e.g., patterns that involve verb phrases in imperative form) have been used [Jung et al., 2010],[Louvigne et al., 2012],[Weber et al., 2012]. User communities [Pareti et al., 2014] and crowdsourcing [Chulef et al., 2001] have been also used to create know-how knowledge.

Most works aim to transform the data into a structured form, like an ontology [Jung et al., 2010], [Pareti et al., 2014], [Ryu et al., 2010], a taxonomy [Chulef et al., 2001], [Strohmaier et al., 2009], [Smith and Lieberman, 2010a] or an activity model [Perkowitz et al., 2004]. To do so, they employ structural information such as HTML tags [Jung et al., 2010],[Pareti et al., 2014], or enumeration [Perkowitz et al., 2004]. However, an existing goal modeling and inference approach can be used, e.g., plan libraries [Smith and Lieberman, 2010b], or rule-based annotators [Strohmaier et al., 2009; Louvigne et al., 2012]. In Chapter 5, we introduce a different technique for extracting actions and goals from the

web site *43Things* (other sites with similar purpose are linkagoal.com, mylifelist.org). Our approach deals with free-form text, where structural information is not available nor are standard expressions, and tries to limit human effort (e.g., hand-crafted rules) as much as possible. The derived knowledge is used to build a novel goal model that we introduce in Chapter 4 and aims to capture the interconnections among actions and goals in order to make recommendations and answer queries. Due to the different needs of the problems we are dealing with (problems in the familly of *finding related items*), our goal model does not perform goal recognition and next action inference (i.e., plan inference) similarly to the existing goal models in the literature that we are presenting in this Chapter.

Moreover, text analysis is important when users describe explicitly their goals using text [Carberry, 1983], e.g., "I want to eat italian food in a restaurant nearby". Goals need to be inferred from the user input. For example, the keyword query such as "Michael Jackson, songs" can be transformed to either "I want to listen to M.J.'s songs" or "I want to download M.J.'s songs" based on analysis of verb-phrases on Web pages or result snippets [He, 2010]. Overall, systems that use user-generated content can benefit from NLP and text mining techniques.

Text Corpus Examples

A plan library in the form of a *hierarchy of goals* is an example of goal model built from text data. The data were taken from a social networking site designed to allow users to share their goals [Smith and Lieberman, 2010b]. The textual descriptions of goals can be analyzed, and connections between goals can be extracted by looking into common verb-phrases. For example, when a similar verb phrase is found in the description of a goal and in a post explaining how the user accomplished another goal, the two goals can be connected accordingly.

Another example of the use of text analysis is the construction of a complete *plan library*, this time based on an existing taxonomy of human goals built by psychologists and sociologists [Strohmaier et al., 2009]. The *taxonomy* contains goals described by verb phrases such as "get married" and "become happy". The knowledge about how a goal can be implemented was derived from verb phrases that co-occur frequently in Web pages with the textual description of the goal.

Another valuable source of data is Twitter. Twitter posts contain extracts related to goals. For instance, messages consisting of motivational messages such as "*Moi-lolita makes me want to learn some french #mangolanguages just to sing along to it.*", "*Getting ready for our trip in France, time to learn some french!*" contain information about learning goals [Louvigne et al., 2012]. Every such message consists of a set of textual features e.g., keywords such as "*because*", "*so that*", "*having*" and a set of conceptual

features. The latter features are motivational factors that reflect the difficulty or the engagement of the user to the respective goal.

To extract knowledge from such a corpus, *statistical models with rule induction* based on natural language processing patterns and other text mining techniques can be used. For example, if we consider a pattern *Verb+Infinitive*, and a phrase derived from a post about Disneyland: "*would like to see the princesses*", the phrase will be first characterized as goal (since it matches the pattern) and then will be disintegrated into: the intention verb "*would like to*", the action "*see*", and the object of the action "*princesses*". Furthermore, a deeper understanding can be achieved by determining, for example, the intention (referred in the specific work as the *level of intention*) of the user towards the goal. The intention inference can be made using classification methods on intention verbs, e.g., "thinking of going" expresses weaker intention than "want to stay". The knowledge derived from this kind of analysis can be exploited from companies for providing better products and services to consumers and for personalized target marketing.

2.2.6 Discussion

The previous subsections have presented generic goal modeling and recognition methods, leaving out unnecessary application-specific details and keeping them under the common prism introduced in Section 2.1. There are, of course, methods that have been intentionally left out of this discussion, e.g., works in the area of planning for stimulating human reactions, mainly because these works focus on non-data management issues, and hence are out of the scope of this study. Figure 2.2 provides a condensed overview of the alternative approaches, splitting goal modeling into its two components: goal data collection and model construction.

Table 2.2 summarizes the goal models focusing on these important features: the *main source* of goal data used for building the model, the *type* of the model, the *observation*, i.e., what is observed for finding the user goal, and the *model elements*, i.e., which goal-related concepts (ref. Sec. 2.1) are captured by the specific model.

Table 2.3 shows the methods used for building different goal model types. The main methods are: (a) *expert analysis*, typically used for goal modeling based on complete records, (b) *statistical analysis* typically used with corpus-based models, (c) *feature discovery and rule definition* in cases of taxonomy-based goal modeling, and (e) *behavior theory selection* in cases of modeling based on behavioral theories. Of course, these methods have been used in other cases too. For instance, expert analysis is also often used in corpus-based models, where experts may determine the structure of a model, e.g., the connections among the variables of an HMM [Hoelzl et al., 2012]. We also often see dif-

Main source	**Type**	**Short description**	**Observations**	**Model Elements**
Complete records	Plan Libraries	set of all plans represented by action sequences, or action hierarchies with links showing post-pre conditions; goals explicitly stated in each plan	actions, or actions and env. variables	actions, or actions and pre/post conditions
Complete records	Consistency graphs	graph structure with nodes: all actions and goals and links representing restrictions on action performance pre/post conditions	actions	actions, goals
Complete records	Action-centric	transition graph (starting from initial state) with two types of levels: levels of observed actions, and level of environment variable states; the links reflect action	states	env. variables, actions
Taxonomies	Taxonomies & classifiers	env. variables(features), functions capturing latent interdependencies among env.variables; a set of goal classes;no predefined set of goals	actions	env. variables, goal classes
Taxonomies	Taxono-mies & annotators	env. variables (features), patterns/rules involving env. variables (possibly with weights); a set of goal classes;no predefined set of goals	actions	env. variables, goal classes
Corpus	N-order Markov	a set of plans represented by action sequences, or a sequence of state transitions leading to a goal;	actions or states	actions or states
Corpus	VOMs	suffix tree for each goal; with nodes: actions that belong to plans for the goal; and links connect actions to get plans of different length	actions or states	actions or states
Corpus	HMMs	graph structures; with nodes observed variables; and links expressing emission probabilities; state transition matrices/functions; goals explicitly stated	observed variables	observed variables, output variables
Corpus	IHMMs	graph structures; with nodes observed variables; and links expressing emission probabilities; state transition matrices/functions; goals explicitly stated	context(e.g., previous pursued goal), observed variables	env. states, env. variables, goals
Corpus	MLNs	graph structure; with nodes: formula predicates; and links: expressing a set of weighted logic formulae involving actions and goals	actions	actions and goals
Corpus	Bayes Networks	graph structure; with nodes: actions, goals and states; and links expressing causality among them	actions, states, goals	actions, states, goals
Corpus	DBNs	graph structure with levels (for different time slots); nodes: actions, goals and states; and links expressing causality among them	actions, states, goals	actions, states, goals
Behavior Theory	Behavior models	graph structure; with nodes: env. variables(i.e., motivational factors) plus intention variable; with interdependency links expressing hypotheses of behavior models	env. variables	env. variables, intention variables

Table 2.2: Goal Model Type Descriptions

ferent methods employed for constructing a model. For example, probabilities have been combined with expert analysis in plan libraries [Geib and Goldman, 2001]. Moreover, machine learning, feature discovery and expert analysis are often met in taxonomy-based approaches [Baeza-Yates et al., 2006; Herrera et al., 2010; Jansen et al., 2008; Lee et al., 2005; Li et al., 2006].

A number of observations can be made regarding the approaches and their use.

Approaches that rely on complete records give a result if and only if there is a single candidate goal (or plan) matching the current observations. Such behavior is strict and may be sometimes considered inflexible. It may also increase the system response time.

Therefore, if goal exploitation is tightly integrated in the system workflow, the system may be blocked or delayed by the inference step. A solution to this problem is to use goal exploitation complementarily, as an add-on component. In this case, the system works properly when no knowledge about goals is available; when the pursued goal of the actor is successfully recognized, it can be exploited to improve the functionality of the system or the results returned to the actor. Moreover, even when additional time is required, if the inferred goal is adequately exploited, the effort required by the user can be also significantly limited reducing the overall needed time.

However, there exist scenarios where the system should only take into account goals that are certain so as not to confuse the user or deteriorate the system operation. The latter is especially true in critical domains such as security. For instance, plan libraries have been used in the domain of web security [Geib and Goldman, 2001]. However, approaches that perform consistency checking are in general not recommended in cases of adversarial recognition since hostile actors are not expected to follow ordinary plans. For these cases, can be used design tools to minimize the maximal number of observations before a goal is recognized, a task known as goal recognition design [Keren et al., 2016b].

Whether the knowledge of goals will benefit a system or not depends on the correctness and completeness of the data used for constructing the model. The creation of the complete records is generally a hard task, first due to the time and effort required by the domain experts, and second, due to the existence of unknown plans. These techniques are more appropriate for problems where the focus is on a small number of goals.

Choosing the right model based on complete records for an application scenario depends on the construction and consistency checking mechanisms used. In plan libraries, allowed actions are combined to form all the possible valid plans for a set of goals of interest. In action-centric approaches, experts follow a different approach for gathering the knowledge. They predefine goals of interest but not actions. The transition graph in action-centric representations is formed while the environment is being perceived. The final plan synthesis is performed during the goal inference making action-centric models

Models	Machine Learning	Text Mining	Behavior Theory Selection	Hypotheses Formulation	Statistical Analysis	Feature/ Rule Definition	Expert Analysis	User studies Annotations
Plan libraries /graphs		[Smith and Lieberman, 2010b]	[Schmidt et al., 1978]	[Kautz, 1991]	[Geib and Goldman, 2001]		[Sadri, 2012] [Carberry, 2001a] [Schmidt et al., 1978] [Kautz, 1991] [Geib and Goldman, 2001] [Hong, 2000]	
Action-centric					[Sun and Yin, 2007]		[Sun and Yin, 2007] [Ramírez and Geffner, 2011]	
Taxon.	[Baeza-Yates et al., 2006; Herrera et al., 2010; He, 2010; Jansen et al., 2008; Lee et al., 2005; Li et al., 2006]	[He, 2010; Strohmaier et al., 2009]			[Hassan et al., 2010]	[Baeza-Yates et al., 2006; Herrera et al., 2010; He, 2010; Jansen et al., 2008; Lee et al., 2005; Li et al., 2006]	[Baeza-Yates et al., 2006; Broder, 2002; Herrera et al., 2010; Jansen et al., 2008; Lee et al., 2005; Li et al., 2006]	[Broder, 2002; Kang and Kim, 2003; Lee et al., 2005; Rose and Levinson, 2004]
N-order Markov	[Sadikov et al., 2010]				[Blaylock and Allen, 2003] [Sadikov et al., 2010]			
VOMs	[Armentano and Amandi, 2009]				[Armentano and Amandi, 2009]		[Armentano and Amandi, 2009]	
(IO) HMMs					[Hoelzl et al., 2012][Gold, 2010]		[Hoelzl et al., 2012][Gold, 2010]	
MLNs					[Ha et al., 2012; Kautz, 1991; Mott et al., 2006]		[Ha et al., 2012; Kautz, 1991; Mott et al., 2006]	
BNs					[Horvitz et al., 1998; Huber and Simpson, 2003; Patterson et al., 2003]		[Horvitz et al., 1998; Huber and Simpson, 2003; Patterson et al., 2003]	
Behavioral Models			[Bagozzi and Dholakia, 2002; Chow and Chan, 2008; Cheung and Lee, 2010; Hsu and Lin, 2008; Parra-Lopez et al., 2011]	[Chelmis and Prasanna, 2012; De Choudhury et al., 2007; Perugini and Bagozzi, 2001]				[Bagozzi and Dholakia, 2002; Cheung and Lee, 2010; Chow and Chan, 2008; Hsu and Lin, 2008; Parra-Lopez et al., 2011]

Table 2.3: Goal Recognition

appropriate for cases where gathering exhaustively all the plans is considered very costly or it is not possible. Automatic consistency checking mechanisms are used while new observations become available to remove inaccuracies and revised judgments made earlier [Yin et al., 2007].

On the other hand, in plan libraries, when a plan does not contain an action that has been observed, it becomes inconsistent no matter if the previous observed actions agree with the plan. One could say that plan libraries include the most common plans that actors may follow in order to achieve some goal. In a closed-world (closed-world assumption), this collection is considered complete. This observation is the reason for both the advantages (certainty and clear knowledge) and disadvantages (static, no new plans) of this model. On the flip side, in consistency graphs, there is no predefined number of plans; every action is connected with every goal unless an inconsistency is caused by this connection. This way, a set of observations that have not been met before can be associated to a goal. But at the same time, the more actions and goals exist in a consistency graph, the more expensive and complicated the inconsistency checking becomes. Furthermore, it becomes challenging to make plan recognition, since the plans are not encoded in the model. Consistency graphs are in general better fit for providing causality relationships between actions and goals.

Approaches that rely on a corpus deal with the problem of gathering goal knowledge by introducing probabilities into the goal models. Action sequences in Markov chains, for instance, could be seen as an incomplete plan library. Since not all the plans are recorded, the latest observations are used as evidence to predict the goal, and by extension the plan, that an actor is following.

There is a high-level connection between hierarchy plan libraries and Variable-order Markov Models (VOMs). They both use a graph representation to encode alternative sequences of actions. In hierarchy plan libraries, this graph representation can reveal whether a goal is pursued by checking whether it is consistent with the environment; in VOMs, it is used to compute the most probable goal.

Variable-order Markov Models (VOMs) and n-order Markov models (chains) handle the actions as black boxes. In order to capture directly the environment variables or state transitions that occur while a goal is being fulfilled, models such as Hidden Markov Models (HMMs) or Bayesian Networks should be employed.

Bayesian networks capture causalities among actions and goals, since goals are a part of the network. Dynamic Bayes Networks allow observing goal and plan evolution over time. HMMs focus on the environment variables and allow the inference of unobserved environment variables from observed ones. In the case of HMMs, the goal is identified by defining the hidden state of the environment. This is why they have been extensively

used in cases where sensors are involved (e.g., activity recognition [Hoelzl et al., 2012]), but they can also be a choice for any environment with variables with latent connections that can reveal the values of other variables.

Latent relationships among environment variables are also captured by *statistical models built on behavioral theories*. These approaches explain and predict user goals (and behavior in general) considering the user as part of a community. The methodology (i.e., the formulation of hypotheses and the performance of statistical analysis to check their validity and reliability) could be exploited for exploring connections of environment variables in different domains as well.

Approaches based on taxonomies that capture latent goals embrace uncertainty, specifically, in the stage of the inference of the goal class to which the actions belong. To introduce goals in the core algorithmic procedures of a system based on a taxonomy, the most important task is the discovery of intrinsic features and properties within the environment that can be used as environment variables. This kind of approaches can be more naturally used by applications such as online recommendations, social media applications, and web retrieval systems, where the environment consists of non-monolithic objects. Such objects can be analyzed and represented in the context of an environment in a goal-aware system. For instance, web pages, forum posts or products in an online store can be analyzed through user studies and experimentation to discover the right features that will be used for modeling environment variables and that can be associated to different (soft) goals. For web pages, we have seen in Subsection 2.2.2 which features have been used (and how) to form an environment in the context of which user goals can be inferred and exploited for effectively performing the retrieval of the web pages by satisfying the latent user goal.

2.3 Goal Exploitation

Given the goal model that captures the information about the operationalization of goals within a system, and the inferred actor's goal, the system may exploit this knowledge to the benefit of the actor (in non-adversarial cases) or itself. In literature, the exploitation of goals has been closely linked to the application scenario. Since the purpose of our study is to bring light to the challenges and the practical value of a goal-aware system, we categorize goal exploitation cases according to the *system behavior* once the user goal is known.

Systems using goal models with full plans, e.g., plan libraries or n-order Markov models, can select which plan will be used to fulfill the goal based on a number of criteria, e.g., computational cost, plan length and so forth. Then, either the system executes the plan automatically, or it guides the user through further interaction towards the fulfillment of

the goal.

Systems that contain goal models where plans or goals are not fully determined (e.g., HMMs or models based on taxonomies) do not replicate a certain plan. The goal can be input to one or more algorithms that support the main functionality of the system. For instance, in web search engines, the inferred goal can be used by the Web Retrieval algorithms to reorder or filter the web sources in the result list.

Overall, we see that goal exploitation can take two forms: (*a*) *exploitation through dynamic environment changes*, where the system provokes changes in the environment that lead directly or indirectly to goal fulfillment while the actor is interacting with the system, and (*b*) *exploitation through system response*, where the system responds to the actor request(s) using algorithms that embrace goals into their core functionality (i.e., algorithms implementing the tasks intended by the system that respond to actor requests considering the inferred goal). Table 2.4 offers a hierarchical organization of the different cases.

2.3.1 Exploitation through Dynamic Environment Changes

We further consider two subcategories of goal exploitation in this category: (*a*) acting in anticipation of the actor's actions, and (*b*) promoting/facilitating actions.

Acting in Anticipation of the Actor's Actions

In this case, the system automatically performs actions (instead of the actor) or it changes the environment state (before the actor performs the actions she has in mind).

By Automatic Action Execution. In principle, every type of application that allows interaction with the user can leverage user goals. A system that "understands" the objects of interest to an actor, and monitors the user operations and interactions in the environment of the system, can automatically change its operation and behavior according to the actor's goal. For instance, it can *take actions on its own by invoking the commands provided by the system interface* (i.e., the commands that the user can perform through the interface) towards the fulfillment of the user goal. Such goal-aware interfaces are known as *intelligent interfaces* and have been used into applications like web browsers, text editors and search engines [Armentano and Amandi, 2009; Lesh et al., 1999; Lieberman, 2009]. Another example are computer or web security systems (e.g., [Geib and Goldman, 2001]), where the system can automatically change its operation to prevent user attacks. In a goal-aware text-editor, for instance, by observing a sequence of menu selections and clicks (i.e., actions), the editor may infer that the user aims to disable auto-correction. To save the user from browsing the various menu options, the editor can directly fulfill the goal (i.e., the deactivation of auto-correction). To avoid misinterpretations, a confirmation

Exploitation Cases		**Indicative References**
Exploitation Through Dynamic Environment Changes		
Acting in Anticipation of the Actor's Actions	By automatic action execution	[Armentano and Amandi, 2009], [Geib and Goldman, 2001], [Lieberman, 2009], [Lesh., 1998]
	By state transitions	[Charniak and Goldman, 2013], [Ha et al., 2012], [Meehan, 1981], [Riedl, 2004], [Schank, 1995] [Han and Pereira, 2010; Roy et al., 2007]
Promoting/ Facilitating Actions	By interface adaptation	[Dragunov et al., 2005], [Armentano and Amandi, 2009], [Lesh., 1998], [Lieberman, 2009]
	By exposing actors to other actors' plans	[Louvigne et al., 2012]
Exploitation Through System Response		
Adjusting the system response	By posteriori adjustment of the initial response of the system	[Broder, 2002], [Lu et al., 2006] [Li et al., 2006], [Herrera et al., 2010]
	By returning information about actions	[Carberry, 1983], [Blaylock and Allen, 2005] [Crook and Lemon, 2010] [Maragoudakis et al., 2007]
Side Services		[Castellanos et al., 2012], [Ajzen, 1991] [Chelmis and Prasanna, 2012], [De Choudhury et al., 2007], [Perugini and Bagozzi, 2001]
Object Modeling		[Carpineto et al., 2009; Sadikov et al., 2010]

Table 2.4: Goal Exploitation

question may be posed to the actor before the system starts performing the actions that will fulfill the goal.

By State Transitions. Apart from goal-aware systems that act proactively in anticipation of the user actions to fulfill user goals, there exist systems that trigger *environment state transitions* when the goal is inferred in order to offer a different user experience, e.g., a more interesting, amusing, or unexpected experience.

A well known example in this category are the *interactive virtual environments or narrative managers*. Interactive virtual environments have been suggested for education and training environments [Mott et al., 2006; Louvigne et al., 2012], as well as for entertainment (game playing) [Gold, 2010; Ha et al., 2012; Kabanza et al., 2010]. In *education*, interactive virtual environments engage students in learning procedures that serve educational purposes being at the same time amusing and pleasant. For example, in a virtual environment for microbiology, the laboratory changes according to the learner's goal, while s/he is trying to resolve a science mystery [Mott et al., 2006].

In *game playing*, narrative managers, depending on the player's goal, change the environment states to introduce unexpected events, e.g., unlock new powers or traps, or prevent the player from repeating the same strategies, i.e., *plans* [Ha et al., 2012]. Such goal exploitation mechanisms *differentiate the player's experience periodically* and enhance the player's loyalty to the game. Games often start with a trial session, during which the game directs players to follow specific goals in order to familiarize themselves with the environment. In this way, the game can keep track of the actions players follow for achieving their goals and can enhance an existing goal model (that expresses *the average player*), or create a model *for the specific player*. The latter can offer a more personalized experience.

Another example of systems that perform state transitions based on the inferred goals of the actors is that of assistance technologies or intelligent homes [Han and Pereira, 2010; Roy et al., 2007]. The practical value of these systems is especially high in social groups that deal with problems such as mobility difficulties, vision or memory problems causing them difficulties in performing everyday tasks. The environment in these cases consists of variables that indicate mainly sensor values: locations, moves of certain body parts, sounds etc. Furthermore, medical or other personal data may be captured. When the actor's goal is inferred, certain environment variable states change before the actor performs any further actions. For instance, the room temperature, the volume of the television, the location of an object may change. The state transitions may directly cause the goal fulfillment, or some additional actions may be expected by the actor. Easiness and effectiveness are the desired qualities for such systems; while personalization elements may be desirable.

Promoting/Facilitating Actions

In this case of goal exploitation, the system somehow suggests actions to the actor by making them more "obvious", and easily accessible. It is, however, up to the actor to perform them.

By Interface Adaptations. To ensure the fulfillment of a goal, a system may facilitate or guide the actor to perform certain actions through various adaptations. These adaptations include the addition of graphics or animations, or the use of vocal input/output means, or the communication via other sensor channels [Lieberman, 2009]. Intelligent interfaces additionally to the automatic execution of actions may perform interface adaptations as well [Armentano and Amandi, 2009; Lesh., 1998; Lieberman, 2009]. Other features of adaptation or personalization are pointing out alternative paths, and reordering or highlighting interface elements related to the user goals. Another important issue that the system needs to deal with is the changes in actor goals and trigger the performance of additional actions whenever is needed.

An example of such a system is TaskTracer [Dragunov et al., 2005], a research tool that consists of a number of intelligent interfaces that were designed to be on top of every desktop activity regarding Microsoft Office applications, Visual Studio and Internet Explorer in Windows XP.

By Exposing Actors to Other Actors' Plans. Another way to exploit goals is by exposing the users to information about the goals of other actors. Specifically, in *educational interactive virtual environments*, exposing learners to information regarding the operationalization of goals of their peers was found very successful [Louvigne et al., 2012]. There is a theory behind this, called *observational goal setting theory*, that suggests that information about goals of others may help the current learner (actor) to stay committed to her/his goals.

2.3.2 Exploitation Through System Responses

We further consider these subcategories of goal exploitation in this category: (a) adjusting the system response, (b) side services, and (c) object modeling.

Adjusting the system response

In contrast to goal exploitation through dynamic changes in the environment, there are cases in which the system does not respond unless the actor makes an explicit request. Practically, the actor's goal is fulfilled through a plan containing actions that become available to the actor via the response of the system. The final selection of the actions

is made by the actor though. The actions that the actor performs after the system's response may optionally be tracked and used to adapt the response. *Web Information Retrieval* is the most well studied application domain.

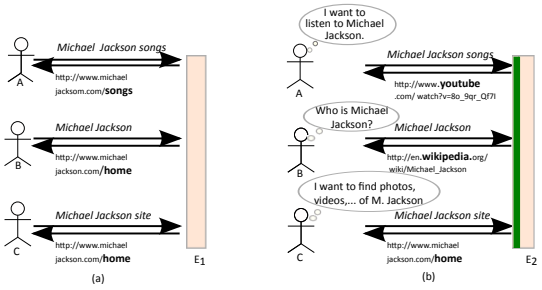


Figure 2.8: Answering web queries using (a) only IR techniques, or (b) in combination with goal recognition and exploitation.

Although the use of search engines has been significantly expanded in the last decades, users may not know exactly what they are searching for, or they may not know how to actually express it in a query language even if the query is expected in a form as simple as a set of keywords. Since users aim at discovering information (i.e., resources) and they keep browsing the results returned from a search, one can consider as an *action* the fact that the user poses a query and sees the results (the change of the information that a user has seen is reflected in the state of the environment) and this task stops when the user finds what s/he needs. If the user is not satisfied with any of the results, she has to pose another query and continue the same process. Thus, a *goal* can be modeled also as a set of environment variable states, and the clicking on the results returned for one or more queries the user has posed can be seen as the *actions*.

Goal-aware information retrieval allows users to cope with information overload and reach faster the information of interest. Clearly, understanding what the user intends to do (ref. Figure. 2.8b) with the data and *promoting results* that help fulfilling that goal can significantly improve the quality of the results, and increase user satisfaction. This has led *search engine* providers into the study of ways to understand what the user had in mind. Over the last two decades, there have been *efforts to make search engines able to recognize the goal of the user*, i.e., what the user wants to do with the retrieved information [Broder, 2002; Herrera et al., 2010], and *act accordingly*. For instance, consider three users that pose the following queries: (1) "*Michael Jackson songs*", (2) "*Michael Jackson*", and (3) "*Michael Jackson site*", respectively (ref. Figure. 2.8a). For the first user, the search engine returns the link *www.michael jackson.com/songs* containing the list of all the songs the artist has sung. For the second and third user, it returns the link to the artist's web page. At first sight, all the results seem to be satisfactory to the users. However, the first

user does not actually want to see the list of songs, but wants to listen to them. Thus, a preferable answer would have been a link to the artist songs on YouTube. The second user was actually interested in learning about Michael Jackson's life, thus, a link to his Wikipedia page would have been a more useful resource than his home page. Finally, the third user is interested in finding photos, videos and filmography of the actor. For that user, the actor's web site is indeed the best resource.

The goal taxonomies of queries presented in Section 2.2 have been plugged into existing web search engines to adapt the retrieved results to the goal behind the current user query. For example, retrieval algorithms that consider goals ensure that queries matching the class of *informational* goals should be answered by diverse web resources that cover the query from different perspectives, contain complementary and controversial information and offer various levels of comprehension. All *these requirements are guaranteed in the respective goals by conditions on environment variables* such as resource diversity, size of resource, etc.

Queries that match the class of *navigational* goals should be answered by a specific web resources that are characterized by perfectness, uniqueness, and authority [Lu et al., 2006]. In this case, users may be certain of the existence of the site because they have accessed it before or they have been informed about its existence by an external resource or they assume it exists. For instance, a user may assume that there is a web site for a scientific laboratory even if s/he has no information about it. In order for a goal-aware engine to detect authoritative sites for navigational queries, it exploits the *inferred goal that contains conditions on environment variables (features)* describing click streams from query logs or on features regarding the url of a resource, such as the length of the longest substring of the query that can be matched to the url [Lu et al., 2006].

Another example is the answering of queries given one query that expresses a *transactional* goal. In this case, goal-aware web search engines answer the queries using a specific fraction of the web collection, since according to studies, transactions can be performed only in a small number of web sites that can be possibly distinguished from common pages. Indeed, there have been efforts to spot "transactional" pages and create a collection of web resources that will be used only for transactional queries. An example is an annotator based on regular expressions and gazetteer look-ups that was built for queries related with two types of user activities on the web: software downloads and entry forms [Li et al., 2006].

By returning information about actions. There also exist systems where after the goal is inferred, the system tries to *return information about the plan, i.e., about the related actions and action preconditions needed for fulfilling the goal.* This is met in dialogue systems for instance, which are computer systems intended to converse with a human with

a coherent structure. They support a broad range of applications in enterprises, education, government, health-care, and entertainment, such as customer care and helpdesk services, technical support, and informational services about news, entertainment topics, the stock market, or any other type of information stored in a knowledge base [Carberry, 1983; Crook and Lemon, 2010; Maragoudakis et al., 2007]. Respectively, in Linux systems have been considered dialogue systems for goals that can be described in a high-level as tasks such as sorting the files and subfolders in a folder, or renaming the files based on a pattern. Such goals are operationalized by command line commands [Blaylock and Allen, 2005]. The dialogue systems first infer lower-level goals, and then gradually build a *complete plan* as the dialogue progresses [Carberry, 1983]. Thus, goal inference is performed hierarchically.

Side Services

Goals can also be exploited *by side services for marketing purposes*, e.g., targeted offers, market analysis and so forth. Online user generated content, offers great opportunities for extracting and encoding knowledge about human goals. For instance, goals expressed in phrases such as *I want to visit France* in Twitter or travel websites constitute valuable information for marketing in traveling and leisure industry [Castellanos et al., 2012].

Several personalized services benefit from the prediction of user behavior in a social platforms [Ajzen, 1991; Chelmis and Prasanna, 2012; De Choudhury et al., 2007; Perugini and Bagozzi, 2001]. For example, in online social virtual games one can meet motivations capturing notions such as "entertaining", or "being challenged by others". A user may be interested in a recommendation of a game that offers entertaining missions (i.e., by trying to satisfy related goals) motivated by a tendency to entertain herself while another user may be willing to buy a game pursue missions of great difficulty motivated by the fact that s/he is challenged by other players.

There are currently no algorithms capturing goal knowledge that manage to perform in an automatic way, i.e., without the interference of a marketing team, tasks such as recommendations in a way that will help users to fulfill their goals, making it a challenging open direction.

Object Modeling

When an object, e.g., a query or a web page, is involved in the fulfillment of one or more goals, either in an action, or in an environment state, or goal, the information about operationalization can be used for the modeling of the object. Two clustering algorithms have exploited such information for object representation. The first is designed

for *clustering the results retrieved by a web search engine for a certain query.* The web pages in the result set are clustered into hierarchies that reflect the different *aspects of the query* [Carpineto et al., 2009] to allow the user to access the results that are only related to a specific goal. Such goal-aware techniques are especially useful in *mobile search* because mobile users are typically not willing to pose more than one query per session nor to scroll through long lists of results, and may be overwhelmed by a large volume of unrelated data. The other algorithm is designed for query clustering in order to suggest to the users those *related* to a query at-hand [Sadikov et al., 2010]. A corpus-based goal modeling approach was followed. The used corpus consists of anonymized logs from Google search engine containing user queries and clicks on pages from the respective result lists within a user session. Query refinements are considered to be the actions, and the documents the potential user goals. A graph for each query is created. The nodes representing the goals are absorbing nodes, i.e., once the user visits a document, it is assumed that s/he stops having the same searching goal in mind unless s/he further refines the same query again. Subsequently, the goal model is constructed; a *Markov chain* model with a transition matrix that reflects the probability with which a user may end up to each goal within a number of steps (actions). Given the model, every query can be transformed into a discrete probability distribution. Finally, a clustering algorithm is performed on these representations capturing the knowledge about goals and intentions of the "average" user.

2.3.3 Discussion

In all the above cases of goal exploitation the goal of the current actor is recognized and exploited to give certain qualities to the usage of the system such as personalization, effectiveness, serendipity and so forth. Effectiveness can have several interpretations; herein, it is used to show that the actor's goal is fulfilled through the system usage. Despite effectiveness that is a quality desired by all goal-aware systems, there exist systems that consider goals to ensure that the actor will accomplish her/his goal easily and as soon as possible, e.g., intelligent interfaces and dialogue systems. Other systems, on the other hand, embrace goals to make the usage of the system (until the goal is satisfied) more interesting, with surprising and pleasant elements or personalized features. These qualities as we have seen are very important for applications such as game playing and target marketing, but at the same time they have been proved beneficial for query answering for instance. In fact, web search engines want to accomplish the right balance between the two; they want to pleasantly surprise the user with their responses and direct her/him in an efficient fulfillment of their goals. Another advantage of goal-aware systems is that the included goal models capture data about the operationalization of goals that allows the

discovery of knowledge during goal exploitation.

The different needs and desired qualities of certain applications can be met by diverse goal model types. Goal-aware systems have used different types of models for the same application domain. For instance, Interactive Narrative Managers have employed Markov Logic Networks [Baikadi et al., 2012], [Ha et al., 2012] [Mott et al., 2006], IOHMMS [Gold, 2010] as well as Bayes Networks and N-order Markov Models [Mott et al., 2006]. Similarly, dialogue systems have employed Markov Models [Maragoudakis et al., 2007] and Bayes Networks [Raux and Ma, 2011].

Qualities	Intelligent Interfaces	Dialogue Systems	Interactive Systems for Education, Game Playing	Personalized target marketing	Web Search Engines	Assistance technologies
velocity	✓	✓	×	×	✓	✓
easiness	✓	✓	×	×	✓	✓
interestingness	✓	×	✓	✓	×	×
personalization	✓	×	✓	✓	✓	✓
serendipity	✓	×	✓	✓	✓	×
extra knowledge	×	×	✓	✓	✓	×
effectiveness	✓	✓	✓	✓	✓	✓

Table 2.5: Qualities added to the usage of certain applications by considering goals.

Chapter 3

Finding Related Posts through Intention-based Matching

In this chapter, *we deal with the problem of finding posts related to a post of interest* (post at hand) in online user communities.

- In particular, we formally introduce a novel method for finding related posts that treats each post as a set of segments and computes the content similarity only across segments of the same intention. Our work focuses on posts in forums within user communities.
- We provide a complete methodology for segment identification and for grouping of the derived segments into intention clusters that exploit the text features' variation.
- We present extensive experiments with real users that confirm the existence of such segments in forum posts of different domains, and verify the effectiveness of the individual steps and decisions of our methodology, including the border selection mechanisms, the selection of features, and last but not least the functions and weights for capturing the text features' variation.
- We describe a fully unsupervised multi-segment ranking technique that provides the top-k forum posts related to a reference post by considering segments with similar intentions and using content similarities within each cluster to derive an overall score between each forum post and the reference post.
- We evaluate the effectiveness of the overall approach on the recommendation of related forum posts using ratings and feedback by users in 3 different domains.

In what follows, we first make a brief introduction to the problem and our approach (Section 3.1), we then present the related work about post matching and text segmentation

(Section 3.2, and a motivating example (Section 3.3) that explains the reasoning behind our solution. We then formally introduce our problem (Section 3.4). In the sequel, we provide a brief overview of the whole approach (Section 3.5) and then we describe the individual steps. First, we describe our segmentation method (Section 3.6), and a way to identify segments of the same intention (Section 3.7). Then, we present our segment-based related forum post finding technique (Section 3.8). Finally, we conclude with a detailed experimental evaluation (Section 3.9).

3.1 Introduction

Forums give the possibility to the users to find solutions and make decisions about problems in a great range of domains like traveling (e.g., *Trip Advisor* forum), health (e.g., *Medhelp*), law (e.g., *ExpertLaw*), computer programming (e.g., *Stackoverflow*), science (e.g., *chemistry* in Stackexchange), technology (e.g., *HP support forum*) and everyday life issues such as parenting (*parenting.stackexchange*). Unfortunately, relatedness that has traditionally been translated into content similarity [Weng et al., 2011; Govindaraju and Ramanathan, 2012] has not been proven very effective in the case of forums because searches are done under specific thematic categories, e.g., *printers, or hotels in New York*, in which the content of all the posts is anyway similar. To deal with this problem, we have introduced a goal-aware selection approach that treats posts instead of monolithic entities as composite objects, each intending to serve a different goal (i.e., having a different *intention*).

Indeed, a forum post consists of parts, each serving a different goal in author's mind, i.e., to communicate a message to the reader through the text. For instance, a part may serve to describe a problem that the author has, another to provide background information in order to put the reader into context, a third to express a desire, and a fourth to reach a conclusion. We refer to these parts of a forum post as *segments*.

The relatedness of two posts can then be based on a comparison across segments that serve the same goal, i.e., they are intended for the same purpose, instead of a comparison of the two posts as wholes.The comparison among text segments can be performed by Information Retrieval methods, such as one of the many TF/IDF or BM25 variants [Robertson et al., 1998] or language-model based methods [Jeon et al., 2005], or using topics generated by topic modeling techniques like LDA [Zhou et al., 2011; Blei, 2012], paraphrasing techniques [Berant and Liang, 2014] or even auxiliary external services [Wen et al., 2015], with the latter been used especially for documents with short and poor content, e.g., tweets. However in our approach, considering that the authors of the forum posts intend to serve different goals, the meaning and importance of a term is estimated based on the

intention of the segment in which the term is found.

Identifying the segments in a forum post is a challenging task. Forum posts are typically one or two paragraph long, with complete sentences. They do not follow the abbreviated style used in micro-blogs, but at the same time, since they are intended for interactive discussions, they are not verbose and they lack the structural constructs (e.g., sections) typically used in full-text documents to identify thematic units. Furthermore, since they are driven by the common needs of the forum participants, they draw heavily their content from a common vocabulary (that depends on the nature/topic of the forum), which means that topic variation, i.e., the used vocabulary, is not a very distinctive factor for the identification of the segments. To deal with this limitation we resort to text features (characteristics) whose variation can identify a passage from one segment to another. We made this choice after realizing that style, tone, brevity, verb tense and other grammatical characteristics that the user chooses to use in the post may serve as indicators of a change in the message that the author is trying to communicate. We refer to these characteristics as *features* and use the term *communication means* (CM for short) to refer to groups of such features. The idea of using communication means for capturing the intention of a segment (or intended message) is analogous to the idea of using keywords to represent a topic. Similar to the way that a variation in a weighted vector of words signals a change in the topic [Hearst, '97; Misra et al., 2009], a variation in a vector of text features signals a change in the intended message.

We have developed a framework for finding related forum posts that is based on the above idea. By exploiting the communication means, the system identifies the different segments within each forum post and splits the forum post into these segments. Segments serving the same intention are identified and grouped together. Given a forum post at hand, its segments are identified and the matching score of each segment with the other forum posts' segments that have the same intention is computed. To compute the segment scores, the used term weighting scheme is adjusted to consider the intention of the segment where the term is found. The segments with the highest individual scores are selected and their scores are combined to compute a score that indicates how the forum post at hand is believed to be related to other existing forum posts, and based on this score we select the top-k posts.

Note that methods that enrich text content exploiting terms, synonyms, latent topics etc. from knowledge bases such as Wikipedia, WordNet, or web search engines [Wen et al., 2015], [Hu et al., 2009], or concept graphs and complex language models [Weng et al., 2011] can still be employed in our method for the comparison among segments. We are not suggesting a new text comparison method, but we propose a method that makes the existing comparison methods more accurate.

3.2 Post Matching and Segmentation Techniques

There exists considerable amount of work for post matching in Question Answering Communities (QAC) where users seek answers to general-interest, factual, or informational questions [Chen et al., 2011]. Apart from computing the explicit content similarities of threads [Jeon et al., 2005; Zhou et al., 2011] such systems may also leverage the syntactic structure of the questions posted in such forums in order to match questions (e.g., [Wang et al., 2009, 2010]) or the thread post-reply structure (e.g., [Singh et al., 2012]). Another approach is to use different combinations of content, semantic, syntactic, and authorship-related features to classify questions as relevant or not [Hong and Davison, 2009; Shtok et al., 2012]. However, in question repositories, posts are plain questions. On the contrary, we suggest a method that enables the use of such techniques on elaborate forum posts that consist of multiple segments. Specifically, depending on how deep one can afford and wants to go into the content similarity, apart from traditional retrieval techniques [Robertson et al., 1998], language model-based methods and semantic text comparisons [Jeon et al., 2005; Zhou et al., 2011], [Berant and Liang, 2014; Wen et al., 2015] could be exploited by our matching technique when the comparison of the text of the segments is performed.

[Segmentation methods] Segmentation methods are divided into 2 broad groups. The first is the *topical segmentation* where adjacent pairs of text blocks are compared for overall similarity based on terms or topics [Misra et al., 2009] or lexical chains [Hearst, '97]. Topic text segmentation is not suitable for our case since we are interested in the author's intention and not the actual topic. The second group of segmentation methods is the *Transcribed oral-discourse* techniques that has been used in the analysis of transcribed oral communication and uses linguistic criteria [Passonneau and Litman, 1993]. These are not applicable to our case which is for written discourse.

3.3 Motivation

Consider a user that identified in a forum site the post A of Figure 3.1 as being of interest, and would like the system to show also other posts that are of interest. Forum post B seems to be such a post since both A and B have a number of important keywords in common (e.g., RAID 0, 320GB, disk drive, HP). However, the fundamental question asked in A is whether performance will degrade ("Do you know ... performance"), while in B is about adding an extra drive ("I am thinking to add ... system?"). Many of the keywords

Doc A
I have an HP system with a RAID 0 controller and 4 disks in form of a JBOD. I would like to install Hadoop with a replication 4 HDFS and only 320GB of disk space used from every disc. Do you know whether it would perform ok or whether the partial use of the disk would degrade performance. Friends have downloaded the Cloudera distribution but it didn't work. It stopped since the web site was suggesting to have 1TB disks. I am asking because I do not want to install Linux to find that my HW configuration is not right.

Doc B
My boss gave me yesterday an HP Pavilion computer with Intel Matrix Storage System, a 320GB drive and Linux preinstalled. I am thinking to add an extra drive using a RAID 0 or 1. Can I do it without having to rebuild the entire system? I have already looked at the HP official web site for how to use a JBOD. But I have not found anything related to it.

Doc C
Extra RAID drives seem to be the solution to my problem but does adding RAID drives requires a reformat and rebuild of the system to improve performance?

Doc D
My HP Pavilion stops working after 15 min of activity. I called our technical department but no luck. Despite the many calls, I did not manage to find a person with adequate knowledge to find out what is wrong. All they said is bring it to up and we will see, which frustrated me. At the end I had the brilliant idea to move it to a cooler place and voila. No more problems.

Figure 3.1: Four posts from a technical forum.

that the two forum posts have in common do not appear in these two parts. For instance, the keyword HP appears in the first part of A ("I have ... disc") and of B ("My boss ... pre-installed"), that are both informative parts intended to communicate to the reader the general context of the author's situation. The keyword HP also appears in the last part of B ("I have looked ... related to it") that simply informs the reader of a related issue. None of these parts is about the main request of the respective forum post. A similar informative role has the keyword RAID at the beginning of A, while in B it has a significant role in the part intended to communicate the author's main request. Thus, despite the content similarities between A and B, B may not be so much of interest to the user. On the other hand, A and C seem to have little content overlap, but it seems that the user may be interested in reading also C, since the main problem discussed in it is similar to the one discussed in A. Finally, D is very different from A in every aspect, consequently, the user would have little interest in reading it.

Thus, in order to identify posts that are likely to be of interest to a user, knowing that a reference post is of interest to him or her, one needs to identify those that are related to that reference post. As the above examples indicate, content similarity can more accurately determine relatedness if focused on parts of the forum posts that play the same role, e.g., to give the context, to describe a wish, to make a request or provide a solution. Instead, if the content similarity is computed across the documents as a whole, the results may be misleading. The main question that needs to be answered here is how these different parts can be identified and how the content similarity can be computed across these parts.

3.4 Preliminaries

Assume an infinite set $\mathcal{T}$ of *text units*. In its simplest form, a text unit is a word, but one can also consider undivided combinations of words, e.g., "New York", as text units.

A *document* d is a finite sequence of text units, and its *cardinality* $|d|$ is the number of text units it consists of. We will use documents to model forum posts, and for this reason we will use the terms "posts" and "documents" interchangeably. Each text unit in a document is identified by its *position*. A *segment* is a finite sequence of consecutive text units in a document, and is identified by the position of its first and its last text unit. For instance, $[n, m]$, with $n<m$, denotes the segment consisting of the text units from the n-th to the m-th position.

A document can be seen as a sequence of non-overlapping segments, the concatenation of which is the document itself. Its division into such a sequence is known as *segmentation*.

Definition 6 *A* segmentation S^d *of a document* d *is a sequence* $(s_1, s_2, \ldots, s_k)$ *of segments such that for every* i=1..$(k-1)$*, the segments* s_i=$[l, j]$ *and* s_{i+1}=$[m, n]$ *are such that* m=$j+1$*, and the textual concatenation* $s_1 \cup s_2 \cup \ldots \cup s_k$ *is equal to* d*. The number* k*, denoted as* $|S^d|$*, is referred to as the cardinality of the segmentation.*

We refer to the virtual point between two consecutive segments as the *border* between these segments. In a document segmentation $(s_1, \ldots, s_k)$, a *border* b_i between a segment s_i=$[l, j]$ and the subsequent segment s_{i+1}=$[m, n]$, is the position m, i.e., the position of the first text unit of the segment s_{i+1}. We will denote by $\mathcal{B}^{S^d}$ the set of borders between the segments of a segmentation S^d. Note that a segmentation S^d can be equivalently represented by its set $\mathcal{B}^{S^d}$. A segment can be as small as a text unit or as large as the document.

By nature, every piece of text is written with a goal in the mind of its author. At the moment of the text construction, the author selects words and text structure that most effectively fulfill this goal. We have experimentally verified the existence of such goals in forum posts (ref. Sec. 3.9.1).

The goal of a piece of text, i.e., a segment, has been written for, may not be explicitly stated, but by the way it is constructed, it is reflected into the characteristics of the text. Thus, monitoring and identifying strong variations in the characteristics of a document will indicate points where the author *intends to* serve a different goal. We use $\mathcal{I}$ to denote the set of all possible intentions and a function int:$\mathcal{U}\rightarrow\mathcal{I}$ that associates every segment to its intention in $\mathcal{I}$. We refer to the text characteristics as *features*, and we will use the term *feature vector* to refer to the values of these features for a segment s. Since there is such a close correlation between the features and the intention, given that the intention is only in the mind of the author, it is natural to identify the intention using text characteristics.

Definition 7 *Given a set F of n features of interest, an* intention *is identified by a feature vector, i.e., a vector of n values, one for every feature of F.*

The idea of using the features to identify intentions is similar to the idea of using terms to identify topics. In the topic detection literature, the topics of the documents may not be explicitly stated but the terms used in the document are an indication of the topic, and based on this observation, a topic has been defined as a vector of terms Hulpus et al. [2013].

We will use the symbol $\sim$ to indicate two highly similar intentions, and the symbol $\not\sim$ to show highly dissimilar intentions. By abuse of expression, mainly for presentation purposes, we may write that two segments have the *same*, or *different* intentions, meaning that they have highly similar or highly dissimilar intentions, respectively, where similarity can be computed using any of the many vector similarity measures in the literature. In the case of two consecutive segments of a forum that have highly dissimilar intentions, we will characterize the border between them as a *deep* border.

Problem Statement. The challenge we propose to address is as follows: given a collection $\mathcal{D}$ of documents, and a reference document d_q, find those k documents in the collection that are most likely to be related to the reference document d_q, i.e., those documents that will most likely be of interest to a user that already considers d_q being of interest. The specific task is referred to as *document matching*.

3.5 Intention-based Matching

To implement a document matching solution for posts, we need to be able to compute some relatedness score, referred to as the *matching score*, of every document in a document collection to a reference document. To do so, we need to compare the reference document and any other document in the collection. It is our position that the relatedness is better assessed by computing a score, not across the content of the two documents as a whole, but across their segments that have the same intention. To achieve this, each document (including the reference document) is first divided into segments of different intentions (*segmentation phase*). The segments are then clustered together (*segment grouping phase*) so that all the segments with the same intention end up together in the same cluster. Each resulting cluster can now be seen as a representative of some specific goal that is different from that of any other cluster. Segments from the same document that may have ended up in the same cluster are concatenated into one, so that there is at most one segment from each document in each cluster (*segmentation refinement phase*). For each cluster in which the reference document has a segment, the segments, and by extension the documents, with the highest scores in the cluster are selected. The score of two

same-cluster segments of two different documents can be seen as the relatedness of the two documents when considering only the specific intention that the cluster represents (*matching with respect to a specific intention phase*). The relatedness (i.e., *matching score*) of the reference document with another document is computed by a combination of their individual intention relatednesses (i.e., respective segment score) across all the clusters (i.e., intentions) considering the segments from the previous phase. Based on the matching score, the top k most related documents to the reference document can be selected (*matching with respect to all intentions phase*).

There are three main challenges in the above steps. The first is how to segment the documents since the intention is not known, neither explicitly stated in the text. The second is how to recognize whether two segments from different (or the same) documents have the same or highly similar intention, in order to be clustered together. The third is how to compute the similarity among segments of the same intention and combine these similarities to form the matching score between the documents. The following sections describe how we cope with each of these challenges.

3.6 Segmentation of Posts

For a document d, there are $2^{|d|-1}$ possible segmentations. Among them, we are interested in the one that is more accurately aligned with the different intentions of the text. Finding the right segmentation is a challenging task Hearst ['97]; Misra et al. [2009]; Salton et al. [1996], for which there is already a large body of work, from segmentation of queries to segmentation of documents Wen et al. [2015]; Hagen et al. [2011]. In these studies, a good segmentation is one where every segment is (i) coherent and (ii) largely disconnected from its adjacent segments. Since our criterion for segmentation is the intention-based, these two properties translate to a segmentation where every segment: (i) conveys a single clear intention; and (ii) this intention is highly different from those conveyed by the adjacent segments. Equivalently, the above criteria call for segmentation with deep borders.

Definition 8 *An* intention-based segmentation S^d *of a document* d *is a segmentation where for any segment* $s \in S^d$*: (*i*)* $int(u_1) \sim int(u_2)$*, for any subsegments* $u_1, u_2 \sqsubseteq s$*; and (*ii*)* $int(s) \not\sim int(s')$ *where* s' *is any adjacent segment of* s*.*

In finding a good intention-based segmentation, there are three challenges: identify the features to use for identifying the intentions, measure the coherence within a segment alongside the depth of the borders of a candidate segmentation, and, select the best segmentation among the candidates. Sections 3.6.1, 3.6.2, and 3.6.3 study these issues.

Tense(CM_{tense})	present	past	future
Subject (CM_{subj})	I/we	you	it/they/(s)he
Style (CM_{qneg})	interrog.	negative	affirmative
Status (CM_{pasact})	passive	active	
Part of Speech(CM_{pos})	verb	noun	adj./adverb

Table 3.1: Features (cells) and Communication Means (rows)

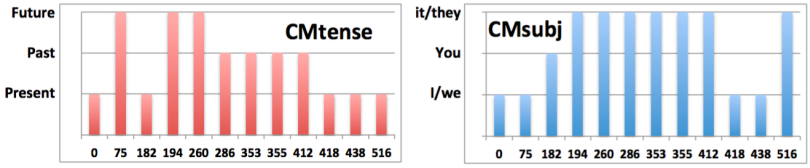
3.6.1 Feature Selection

First of all, we need to decide the features to use for identifying intentions. Content-based features, e.g., terms or keywords, have been used in the past for segmentation [Hearst, '97; Misra et al., 2009]. Keywords have been also used by LDA for topic discovery. Since forum posts are relatively short, they tend to be very concise, which means that the basic keywords are used all over the post, making it hard to identify large topic variations. Another type of features is the discourse-based features, such as pauses or voice stress, that are related to transcribed oral communication [Passonneau and Litman, 1993], but are hard to exploit in text documents such as forum posts. Since posts are written to initiate or continue a discussion, they highly reflect the users' way of communication. Thus, it is natural to consider as features language characteristics that are related to syntax and grammar. The intuition is that a change in the expression style signals a change in the intention. For instance, when an author switches from the first to the third person, that is a signal of a change in the intended message.

Examples of grammar features are the verbs in some specific tense, the passive verbs, the references in the first person, etc. We classify the features into types, referred to as *communication means*. An example of a communication mean (CM) is the `Subject` that contains the features corresponding to references in the first, second and third person. In this way, each CM can be seen as a categorical variable and the features in the CM as its domain. For instance, the CM `Tense` can be seen as a categorical variable that takes the values `past`, `present` or `future`. Table 3.1 illustrates a number of features grouped under their respective CM. Each row in the table corresponds to a CM and each cell to a feature. One can monitor the value of a CM throughout a document (or segment).

Example 5 *The top part of Figure 3.2 illustrates forum post A of Figure 3.1, where the words indicating a value of CM_{subj} are in bold and those indicating a value of CM_{tense} are underlined. The boxes indicate certain positions in the document. Below the text, there are two bar charts that show the values of CM_{tense} and CM_{subj} throughout the document. The x-axis is the position in the document and the y-axis is the categorical value of the variable. In these bar charts, it can be seen that there is a shift in the value of the*

0 **I** have an HP system with a RAID 0 controller and 4 disks in form of a JBOD. 75 **I** would like to install Hadoop with a replication 4 HDFS and only 320GB of disk space used from every disc. 182 Do **you know** whether 201 **it** would perform ok or whether the partial use of the disk 259 would degrade performance. 285 **Friends** have downloaded the Cloudera distribution but 338 **it** didn't work. 355 **It** stopped since 371 **the web site** was suggesting to have 1TB disks. 418 **I** am asking because 436 **I** do not want to install Linux and then realize that 488 **my hardware configuration** is not the right one. 535



	Possible Segmentations
Boxes	75, 182, 201, 259, 285, 338, 355, 371, 418, 436, 488, 535
(*a*) CM_{tense}-Based	([0,75], [76,182], [183,201], [202-285], [286-418], [419-535])
(*b*) CM_{subj}-Based	([0,182], [183,201], [202,418], [419,488], [489,535])
(*c*) CM_{qneg} Shift	([0,182], [183,201], [202,438], [439,535])
(*d*) Intention-Based	([0-182], [183,418], [419-535])
(*e*) Thematic	([0-49], [50-535])

Figure 3.2: CMs and Segmentations.

categorical variable, i.e., the CM. For instance, for CM_{tense} this takes place in positions 75, 182, 201, 285, and 418. Assuming that the time is a strong factor that can signal by itself a change in the author intention, the post can be segmented into the segments shown in line (a) in Figure 3.2. Line (b) shows a segmentation based on the points where there is a change in the CM_{subj} value, and line (c) based on CM_{qneg}. $CM_{pasactive}$ is not present in the post. The segmentations (d) and (e) are discussed in Example 6.

Each CM, or combination of CMs, can be used to define a possible segmentation. We have experimented with different alternatives, either single CMs or combinations thereof. Another important factor is the domain of the categorical variables. For instance, CM_{tense} can have as a domain the $\{past, present, future\}$ or $\{past, not{-}past\}$. To select the best combination, we need to evaluate the effectiveness of each choice. To do so, we measured the diversity of the various segments in a segmentation and compared it to the diversity of the unsegmented post. For measuring the diversity, we use the metrics described in the next section. We note that the features and the CMs that were found to be the best choice are those contained in Table 3.1.

3.6.2 Coherence and Depth Computation

Intuitively, as hinted earlier, to evaluate the quality of a segmentation we need to measure what variation is observed within a segment in terms of the user intentions and how the intentions of a segment differ from those of the adjacent segments (which would justify why the adjacent pieces of text have been placed in different segments). Thus, given a set of features, we need to be able to measure the coherence of a segment and the depth of a border.

Having a coherent segment means that in general we do not want to see large variations across the features observed in the segment, i.e., across the CMs' categorical values that have non-zero appearances. This is a measure known as *evenness* in statistics. Of course, if we select very small segments, there will be very few factors with a non-zero value. Due to the limited segment length, these values will be very similar, hence such segments will be highly coherent, yet, not really useful. To avoid this, in addition to evenness, we also need to consider the number of non-zero features, called *richness*.

The *diversity indices* consider both richness and evenness by measuring how many features have non-zero values, and at the same time how evenly are distributed among features. The value of a diversity index increases when richness and evenness increase, while decreasing in any other case.

To estimate diversity, we represent every communication mean CM_r by a distribution table (i.e., a vector) DSb_{CM_r}. Intuitively, each distribution table corresponds to a row of Table 3.1. The value of the element j of the table DSb_{CM_r}, denoted as $DSb_{CM_r}[j]$, indicates the number of times the value in column j of the CM r appears in the segment. For instance, a $DSb_{CM_{tense}}$ equals to $[2, 3, 0]$ means that the segment has 2 verbs in present tense, 3 in past tense and none in future tense. A well-known diversity index is Shannon's index,

$$div_{CM_r}(s_i) = -\sum_{j=1}^{|DSb_{CM_r}|} \frac{DSb_{CM_r}[j]}{All} * log(\frac{DSb_{CM_r}[j]}{All}) \tag{3.1}$$

where $All = \sum_{l=1}^{|DSb_{CM_r}|} DSb_{CM_r}[j]$.

The diversity values of each of the CMs in a segment s_i can be combined together to form a value for its *coherence*, which for a segment s_i can be computed by the following $coh(s)$ function that for categorical variables with at most three values takes value types less than one. (Note that higher diversity means less coherence.)

$$coh(s_i) = \frac{1}{|CM|} \sum_{r=1}^{|CM|} 1.0 - div_{CM_r}(s_i) \tag{3.2}$$

To measure the "depth" of a border, one can exploit the concept of coherence. A border is "deep" if the CMs in the two segments it separates are significantly different. To measure this difference, we remove the border, which in practice would mean that the

segments on its left and right would become a single large segment, and we measure the coherence of this segment. That large "hypothetical" segment will have either a lower coherence than the two individuals (indicating a deep border) or a higher coherence, indicating a shallow border. Thus, the depth of a border b_i between segments s_i and s_{i+1} is:

$$depth(b_i) = \frac{|coh(s_i) - coh(s)| + |coh(s_{i+1}) - coh(s)|}{2 * coh(s)} \tag{3.3}$$

where the segment s is the segment resulting from the concatenation of s_i and s_{i+1}.

In previous work, the distance metrics of cosine dissimilarity, Euclidean distance, and Manhattan distance on term-based representations, have been used to decide whether two segments should remain separated or should be better merged as one. However, in the experiment section, we illustrate that term-based representations and distance metrics are not very effective for intention-based segmentation.

3.6.3 Border Selection

To find the best segmentation we need to select the best border positions in the document. With the ability to measure coherence of a segment and the depth of a border, we can define a measure to judge how strong or weak a border position is. A possible border b_i in position i is a good choice if each of the two segments s_i and s_{i+1} that b_i separates has a strong coherence and b_i has a high depth. Based on this, we assign a score to a possible border position. The score can be computed using a weighted sum of the coherence and depth, the *f-statistics* [Bossart and Prowell, 1998], or any other metric as long as it is consistent with the above principle. We are actually computing it as the average of the three parameters, i.e.,

$$score(b_i) = (coh(s_i) + coh(s_{i+1}) + depth(b_i))/3 \tag{3.4}$$

There are two broad approaches to identify the borders that define an intention-based segmentation in a document. One is a top-down approach that initially considers the whole document as one segment and checks for possible positions a border can be placed in order to split the segment into two. The position is selected so that the resulting two segments have an average score that is better than the score of the borders before the split. The approach recursively splits segments as long as such borders can be found. Its main limitation is that the comparison of the depth and coherence in segments that differ significantly in terms of length may mislead the algorithm. For similar reasons, comparing two long segments may lead to incorrect decisions.

The other approach is bottom-up. It initially considers every text unit as a segment and iteratively merges consecutive segments to form longer segments. The merging of two

consecutive segments is performed by simply removing the border that separates them. We propose different strategies to implement the bottom-up approach. Each strategy uses some different criteria for deciding whether to merge segments or not.

The *first strategy*, referred to as *Tile*, has also been used in thematic segmentation [Hearst, '97]. It iteratively passes through the whole document, and at the end of each iteration, it removes the borders that have a score smaller than a threshold. This threshold is defined as the mean score value of all the present borders but adapted by the standard deviation. This way after each iteration the score of the remaining borders increases (or remains unchanged). The process stops when no border satisfies the criterion.

The *second strategy*, referred to as *StepbyStep*, visits the borders in order, from left to the right. For each border it visits, it checks the coherence of the segment on its left. If that coherence is lower than the coherence of the whole document, the border is deleted and the segments before and after it become one. The algorithm continues until it has visited all the borders. The borders that have not been eliminated at the end specify the final segmentation.

The *third strategy* is referred to as the *Greedy*. It makes multiple passes over the document, and in each pass, it removes only one border, in particular the one with the worst score, which should also be less than some specific threshold. The algorithm stops when there is no border that can be removed either because there are no more borders or because there are no borders with a score less than the threshold. The algorithm makes locally optimal decisions, which means that it may be misled by the diversity of a single CM feature to the overall optimal solution. To avoid this, Greedy is run multiple times, one for each single CM and instead of removing the borders that the algorithm suggests to remove, it marks them for removal. After the step has been repeated for each of the CMs, those borders that have been marked for removal for the most of the times are those that are actually removed. Greedy has a higher execution time comparing to the other two mechanisms due to the multiple passes, but as we will see in the experimental section, it best approximates human-generated segmentations.

Example 6 *Considering the features indicated in Table 3.1, the coherence and depth as defined in Section 3.6.2, and the score of Equation 3.4 the intention based segmentation of the post of Figure 3.2 is the one shown as (d) in Figure 3.2. For comparison, the figure also shows as (e) the thematic segmentation generated by running Hearst's thematic segmentation method on the post [Hearst, '97], which highlights the significant difference between thematic and intention-based segmentation.*

3.7 Segment Grouping

The next step in intention-based post matching is to recognize segments that are intended for the same goal. We actually need to create groups such that segments with similar intentions end up in the same group and segments with different intentions in different groups. Since the actual intention is not known but we have modeled it through a vector of features, a natural choice for creating the desired groups is to perform clustering on the respective feature vectors of the segments. Each cluster can then be seen as a representative of some communication goal. We use I to denote a cluster, and $\mathbb{C}$ to denote the set of the generated clusters.

We have found that using the feature vector as it is (meaning with the absolute values of the features) is not very effective. Instead, we need to capture the relative contribution of each feature, thus we have created a vector of weights that are based on the feature values. We denote this vector with the letter F.

The Weights. We consider two types of weights that capture the strength of the use of each CM categorical value, i.e., of each feature. The *first type* measures the strength of the use of each CM value *within the segment*, i.e., in comparison to the frequency of the other categorical values of the same communication mean appearing in the segment. Using the notion of the distribution table DSb_{CM_r} of a communication mean CM_r introduced in Section 3.6.2, we define the vector F_s of weights, one weight for each feature.

Doc A, Seg 1: I have an HP system ... from every disk
Doc B, Seg 1: My boss gave me ... Linux pre-installed
Doc A, Seg 2: Do you know whether ... have 1TB disks
Doc B, Seg 2: I am thinking to ... the entire system?
Doc A, Seg 3: I am asking because ... the right one
Doc B, Seg 3: I have already looked ... related to it.

Figure 3.3: Segments of forum posts A and B of the Figure 3.1. Segments found to belong to the same intention cluster appear together.

The weights for a segment s are computed according to the formula: $\forall i = 1..|CM|,\ \forall j = 1..|DSb_{CM_r}|$

$$F_s[i * |DSb_{CM_r}| + j] = \frac{DSb_{CM_r}[j]}{\sum_{k=1}^{|DSb_{CM_r}|} DSb_{CM_r}[k]} \tag{3.5}$$

In the above formula, $|CM|$ indicates the number of different CMs we consider. For simplicity, it also assumes that all the CMs have the same number of categorical values, i.e., in the case of Table 3.1, that would be that all the CMs have 3 possible categorical values, but this may not always be the case (see for instance CM_{pasact}.)

The weight $\frac{DSb_{CM_{subj}}[2]}{\sum_{k=1}^{3} DSb_{CM_{subj}}[k]}$, for instance, of the 2^{nd} value of the CM: CM_{subj}, will measure how much stronger the use of the 2^{nd} person is as opposed to the 1^{st} or 3^{rd} person.

The *second type of weight* is derived from a normalization of the absolute number of occurrences of the CM categorical value *across the entire post*. For a specific categorical value, it captures the portion of the overall appearances in the whole document that correspond to the examined segment. Similarly to the weights of the first type, the vector F_s of all the weights of the second type of a segment s is computed according to the formula: $\forall i = 1..|CM|,\ \forall j = 1..|DSb_{CM_r}|$

$$F_s[i * |DSb_{CM_r}| + j] = \frac{DSb_{CM_r}[j]}{DSb^*_{CM_r}[j]} \tag{3.6}$$

where DSb^* denotes a distribution table that considers the whole document as a single segment. As an example, consider a document where we find five verbs in past tenses (CM_{tense}-Value 2), four of which are in the same segment. Then, the weight of this value of CM_{tense} will be high indicating for the value a significant role in the segment.

The *vector representation of each segment* is the concatenation of the two vectors corresponding to the two types of weights. Using the CMs of Table 3.1, the vector will have 28 elements (2 for each feature, corresponding to the two types of weights that were just introduced). Any of the well-known clustering techniques can now be applied on the weight vector representation of the segments.

We have experimented with different clustering algorithms. However, since the reason we employ clustering is to capture common patterns in the use or better the distribution of Communication Means within the segments and within the respective posts, the use of the local density of points to determine the clusters is more natural than the distance between them. The DBSCAN [Ester et al., 1996] algorithm, has been a good choice because: (1) it does not require to know the number of clusters in the data a priori, as opposed to distance-based clustering such as k-means, (2) it can find arbitrarily shaped clusters, and (3) it has a notion of noise.

Segmentation Refinement. It is possible that more than one segment from the same document end up in the same cluster, if they have the same intention but are not consecutive in the document, or the border selection mechanism kept a border between them due to local optimal values of segment diversity and border depth. We make one more pass over the clusters and if such cases are found, all the segments that belong to the same document in a cluster are concatenated into one. In other words, assuming the clustering $\mathbb{C}$ of the segments of a collection of documents $\mathcal{D}$, for every cluster $I \in \mathbb{C}$, a new set of segments is considered instead that is constructed as: $\{s |\ \exists d \in \mathcal{D}\text{: } \bigcup_{s' \in I\ \wedge\ s' \in S^d} s'\}$, where the symbol $\cup$ on segments indicates concatenation. As a result of this step, each document may have at most one segment in each cluster.

Example 7 *Figure 3.4 illustrates the results of the clustering of the segments of all the documents in the forum post dataset HP Forum (described in Section 3.9) from which the 4 documents of Figure 3.1 were taken. The rows correspond to elements of the feature vector. In white are the elements of the first type (Equation (3.5)) and in gray those of the second type (Equation (3.6)). Each of the columns I corresponds to a centroid of the clusters that the clustering produced, i.e., to an identified intention. Figure 3.3, on the other hand, shows which of the segments of the forum posts A and B of Figure 3.1, have been clustered together, i.e., they have been assigned to the same intention.*

CM - Feature	**Feature Vector**	**Intention Cluster Centroids**				
		All	I_0	I_1	I_2	I_3
CM_{tense}-Present	Fs[1]	0.26	0.45	0.10	0.13	0.86
CM_{tense}-Past	Fs[2]	0.16	0.03	0.07	0.82	0.14
CM_{tense}-Future	Fs[3]	0.07	0.07	0.09	0.02	0.04
CM_{subj}-I	Fs[4]	0.22	0.29	0.12	0.28	0.47
CM_{subj}-You	Fs[5]	0.01	0.01	0.02	0.01	0.02
CM_{subj}-She/They	Fs[6]	0.25	0.39	0.10	0.26	0.80
CM_{qneg}-Interrog	Fs[7]	0.05	0.08	0.02	0.02	0.16
CM_{qneg}-Negative	Fs[8]	0.10	0.20	0.05	0.05	0.25
CM_{qneg}-Affir/ve	Fs[9]	0.26	0.36	0.11	0.30	0.85
$CM_{passact}$-Passive	Fs[10]	0.07	0.12	0.04	0.07	0.15
$CM_{passact}$-Active	Fs[11]	0.27	0.38	0.11	0.28	0.87
CM_{pos}-Verb	Fs[12]	0.27	0.39	0.11	0.27	0.88
CM_{pos}-Noun	Fs[13]	0.27	0.37	0.12	0.28	0.86
CM_{pos}-Adverb	Fs[14]	0.25	0.37	0.10	0.28	0.77
CM_{tense}-Present	Fs[15]	1.95	3.39	1.21	1.00	4.19
CM_{tense}-Past	Fs[16]	0.52	0.17	0.39	1.84	0.32
CM_{tense}-Future	Fs[17]	0.11	0.10	0.14	0.03	0.06
CM_{subj}-I	Fs[18]	0.75	0.96	0.59	0.88	1.05
CM_{subj}-You	Fs[19]	0.02	0.02	0.03	0.01	0.02
CM_{subj}-She/They	Fs[20]	1.77	2.67	1.09	1.88	3.45
CM_{qneg}-Interrog	Fs[21]	0.06	0.09	0.02	0.02	0.19
CM_{qneg}-Negative	Fs[22]	0.17	0.30	0.10	0.08	0.34
CM_{qneg}-Affir/ve	Fs[23]	2.69	3.73	1.83	3.04	4.78
$CM_{passact}$-Passive	Fs[24]	0.10	0.17	0.0	0.10	0.19
$CM_{passact}$-Active	Fs[25]	3.21	4.59	2.13	3.41	5.96
CM_{pos}-Verb	Fs[26]	4.00	5.86	2.65	4.05	7.46
CM_{pos}-Noun	Fs[27]	7.17	9.91	4.58	7.52	14.92
CM_{pos}-Adverb	Fs[28]	2.10	3.03	1.43	2.23	3.72

Figure 3.4: Derived Intention Clusters after Segment Clustering

3.8 Matching

To perform document matching, i.e., to identify the documents in a collection that are related to a reference document d_q, one way is to see the document d_q as a query and then measure the relatedness of each other document d' to that query in a way similar to how IR techniques work. As already mentioned, our position is that such a task should not consider each document as a whole but should be specialized on each intention individually, and then combine the results.

Matching with respect to a specific Intention. Each cluster is the projection of every document on the specific intention that the cluster represents. Thus, to measure the *relatedness* of a document d' to the reference document d_q *with respect to a specific intention* I, it is enough to measure the relatedness of the respective segment s' of d' in the cluster I, to the respective segment s_q of d_q in that same cluster.

For computing this relatedness any text comparison, e.g., paraphrasing[Berant and Liang, 2014], language models [Jeon et al., 2005; Zhou et al., 2011], or IR techniques may be employed. One of the most well-known IR techniques is the TF/IDF. In the core of the original TF/IDF method and its probabilistic variance BM25 is the *term weighting scheme* that weighs a term in a document considering the number of its appearances in that document in relationship to the number of its appearances in all the other documents. We devise a version that is somewhere between the original and the BM25, and takes into consideration the intentions. In particular, we start with a variance of TF/IDF that comes close to BM25 and has been implemented in MySQL 5.5.3 for full-text searching. That variance computes the weight of a term t in a document d' as

$$w(t, d') = \frac{log(f_{d'}(t)) + 1}{\sum_{\forall t' \in d'} (log(f_{d'}(t')) + 1) * NU(d')} \tag{3.7}$$

where $f_{d'}(t)$ is the frequency of a term t within the *document* d', and $NU(d')$ is the document length normalization factor that penalizes d' if the number of unique terms in the document is larger than the average number of unique terms across all the documents.

We extend the above formula in a way that the weight of a term is based on the segment it belongs (instead of the document) and the intention (i.e., cluster) that the segment has been assigned to. In particular, the weight of a term t in a segment $s' \in I$ is:

$$w(t, s') = \frac{log(f_{s'}(t)) + 1}{\sum_{\forall t' \in s'} (log(f_{s'}(t')) + 1) * NU(s', I))} \tag{3.8}$$

where $f_{s',I}(t)$ is the frequency of the term t within the segment s', and $NU(s', I)$ the segment length normalization factor that penalizes s' if the number of its unique terms is larger than the average segment length in that intention cluster I. With this approach,

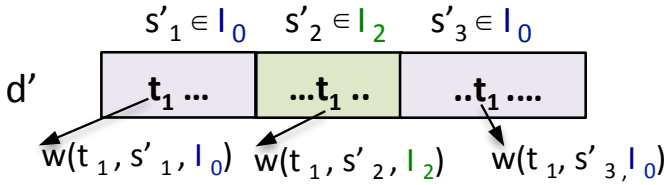


Figure 3.5: Weighting for the same term in different intention clusters.

we generate weights for the terms that may be different for the same term across different segments.

Example 8 *Figure 3.5 illustrates our weighting approach. In a document d', with $S^{d'}$=$\{s'_1$, s'_2, $s'_3\}$, term t_1 is weighted differently when found in segment s'_1 than in segment s'_2 or s'_3. For instance, since s'_1 has been assigned to intention I_0, the weight of term t_1 is based on the terms in s'_1 and those in all the other segments in I_0.*

The relatedness of a document d' to a reference document d_q with respect to an intention I, can now be computed based on the term weights. If s_q and s' are the segments of the documents d_q and d', respectively, in the intention cluster I, the relatedness is:

$$scr(d_q, d', I) = \sum_{\forall t \in s_q} f_{s_q}(t) * w(t, s') * \frac{log(|I| - |I^t|)}{|I^t|} \tag{3.9}$$

where $f_{s_q}(t)$ denotes the frequency of the term t in the segment s_q, $|I|$ the cardinality of the intention cluster, and $|I^t|$ the number of segments in the intention cluster I that contain the term t. The fraction $\frac{log(|I|-|I^t|)}{|I^t|}$ is actually the traditional *probabilistic inverse document frequency*, adjusted for the case of intentions. Moreover, in an application scenario where some clusters are more important than the others, different weights can be considered for each cluster turning the above sum into a weighted sum.

Note that if one of the documents d_q or d' has no segment in the intention I, then the relatedness score is by default 0.

Let $M_I(d_q)$ denote the top-n most related documents to the reference document d_q for the intention I as identified by the relatedness score. Furthermore, let $\mathcal{M}$ denote the set of all such lists for the different intentions. Note that instead of considering the top-n documents for each intention, one could consider only those that are above a specific threshold [Fagin, 1996], however, to be fair across all the intentions that a document contains, we opted for the top-n approach. Algorithm 1 illustrates the above steps.

Matching with respect to All the Intentions. The top-n lists generated across the different intentions, i.e., the set $\mathcal{M}$ mentioned above, are used to generate the k most related documents to the reference document d_q. A new list R is created that contains

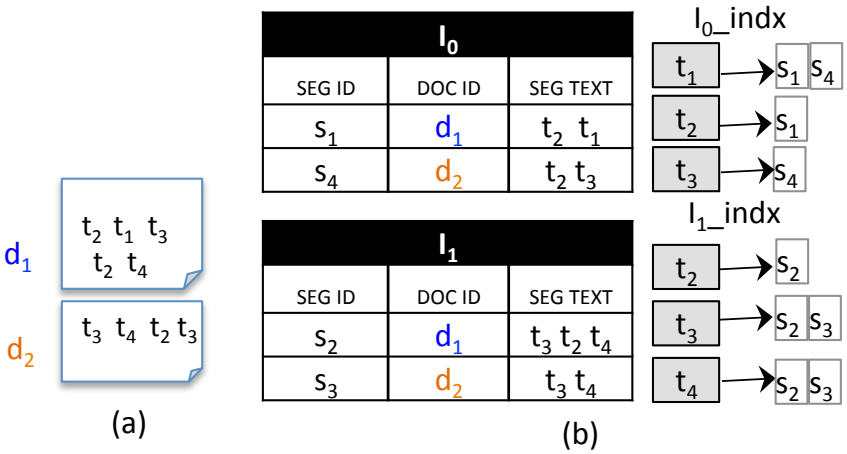


Figure 3.6: On the left, a document collection $\mathcal{D}$={d_1,d_2}. On the right, $\mathcal{D}$ after the segmentation, segment grouping, and indexing step.

every document that appears at least in one of the lists in $\mathcal{M}$. A score is associated to each such document that is the sum of the scores with which this document appears in the various lists in $\mathcal{M}$. The k elements in R with the highest score are returned as answer to the request of the matching documents to the reference document d_q. These steps are indicated in Algorithm 2.

It is important to note that a relatively small value for n (compared to the value of k) will favor documents that have high score in one list in $\mathcal{M}$ even if they do not appear in others, penalizing at the same time documents that may appear in many lists but with lower scores. A relatively high value for n compared to the value of k, on the other hand, will favor documents that appear in many lists even with not very high scores. We have empirically found that a good choice is an n equal to $2 * k$.

Indexing. In contrast to segmentation and segment grouping that are performed offline (pre-processing steps of the document collection), document matching, i.e., the retrieval of the top-k documents for a document query d_q, can be performed online due to its low response time (less than 3 millisecs for a collection with more than 1.5M posts, ref. Section 3.9.2). In practice, in order for Algorithms 1 and 2 to be able to generate fast the (initial) top lists in each cluster I and subsequently generate the final list, we built a full-text index on the terms of all the segments of each segment group (cluster) I. Therefore, we are building $|\mathcal{C}|$ fulltext indexes. In addition, we are building an index on the ids of the documents where the segments belong so as to be able to access faster the segments of a document query d_q. Figure 3.6 graphically illustrates the two clusters (I_0, I_1) and the corresponding indexes (I_0_indx, I_1_indx) that have been formed after the segmentation and segment grouping of a small document collection (d_1, d_2).

Algorithm 1 Single Intention Matching

Input: Cluster I, Doc. Collection $\mathcal{D}$, Document $d_q \in \mathcal{D}$, Int n
Output: List of n documents and their intention matching score

$M_I \leftarrow \emptyset$
for each $s_q \in S^{d_q}$
  if $s_q \notin I$ **continue**; // See footnote[1]
  $scr \leftarrow 0$
  for each $s' \in I$
    $d' \leftarrow \{d \mid s' \in S^d\}$ // See footnote[2]
    for each $t \in s_q$
      $scr \leftarrow scr + f_{s_q}(t) * w(t, s') * log(|I| - |I^t|)/|I^t|$
      $M_I \leftarrow M \cup \langle d', scr \rangle$
Return $\{\langle d', scr \rangle \mid \langle d', scr \rangle \in M_I \wedge scr \in$ top-n scores in $M_I\}$

Algorithm 2 All Intentions Matching

Input: Document Collection $\mathcal{D}$, Document $d_q \in \mathcal{D}$, Int k, n
  Intention Clusters $\mathbb{C}$
Output: List of documents

$L \leftarrow \emptyset$, $M \leftarrow \emptyset$
for each $I \in \mathbb{C}$
  for each $s_q \in S^{d_q}$
    if $s_q \notin I$ **continue**
    $M_I \leftarrow$SingleIntentionMatching(I,$\mathcal{D}$,d_q,n)
    $L \leftarrow L \cup \{M_I\}$
for each $M_I \in L$
  for each $\langle d', scr \rangle \in M_I$
    if exists $\langle d', x \rangle \in M$, with $x \in \mathbb{R}$
      $M \leftarrow M \cup \langle d',\ scr \rangle$
    else $\langle d', x \rangle \leftarrow \langle d',\ x + scr \rangle$
Return $\{d' \mid \langle d', scr \rangle \in M \wedge scr \in$ top-k scores in $M\}$

[1] Due to the segmentation refinement step, there will be only one segment for which $s_q \in I$, and at most one document for which $s' \in S^d$ and $s' \in I$.

3.9 Experimental evaluation

We have evaluated all the steps of our method on the recommendation of related posts i.e., the segmentation, the identification of segments with the same intention and the comparison of the posts based on the similarity across segments of the same intention. We first needed to see whether the segmentation task we perform makes sense. Section 3.9.1 verifies the existence of segments in forum posts; while Section 3.9.1 presents the findings of the evaluation of the segmentation step of our approach contrasting alternative features, border selection mechanisms and coherence/depth functions. In the sequel, we have evaluated our overall approach comparing its effectiveness, in terms of precision, to two baseline methods that are not using any segmentation and our approach when for the segmentation and grouping are used methods other than the intention-based (ref. Section 3.9.2). Moreover, the performance/efficiency of our approach has been evaluated with experiments on data of different sizes (Section 3.9.2).

Datasets. We used three real datasets of posts from forums in three different domains. The first had 111K posts from a product *support* forum (HP Forum, http://h30434.www3.hp.com), with an average post size of 93 terms with 2.3% unique terms (stop-words were not considered). The second dataset, had 32K posts of hotel reviews from a *travel* forum (TripAdvisor) [Ganesan and Zhai, 2011]. The average post size was 195 terms with 3.2% unique content terms. And the third dataset was a dump of a well-known computer *programming* forum (StackOverFlow, http://stackoverflow.com) consisting of 1.5M (it actually consists of 4M posts but we have considered only those with an accepted answer). The average post size was 79 terms with 2.5% unique terms. In all datasets, the number of posts refers only to root posts (i.e., posts that trigger a thread); answers are not included. The percentage of unique terms verifies that in forums since users deal with issues under specific topics, the used vocabulary is limited.

Implementation. For the experiments we used an Ubuntu 0.14.04.1 machine, with 125GB memory, CPU 172 MHz and MySQL 5.5.3. The code was written in Java 1.7.

3.9.1 Segmentation Evaluation

We conducted a user study to: (*i*) validate our observations that posts, despite their relative short size and informal writing style, *can be naturally divided into parts, with each reflecting a different message*; therefore there is a strong user agreement (ref. Section. 3.9.1.A), and to understand *which are the different messages that the authors convey* (ref. Section 3.9.1.B), and (*ii*) to evaluate the automatic segmentation approach (features, border selection mechanisms, coherence/depth functions) (ref. Section 3.9.1). For

	HP Forums	*TripAdvisor*
Offset	*Fleiss's κ/Agreement Percentage*	
±10 chars	0.20/64%	0.35/71%
±25 chars	0.41/71%	0.44/75%
±40 chars	0.68/77%	0.71/83%

Table 3.2: User agreement on the segmentation task

contrasting the alternative features and functions, we consider multWinDiff error; while for the border selection mechanisms we also present how the number of borders and segment coherence is affected by each of the mechanisms. Specifically for this study, we used a randomly selected sample of two of the datasets: 500 posts from the support and 100 posts from the travel forum.

Human Annotation Task. We had 30 participants from five countries that were all computer literate and fluent in English. All the participants had at least a bachelor's degree. Among them, there were users with PhD and PhD candidates in computer science or engineering as well as software developers and engineers. The participants were asked to read each post carefully and *divide it into coherent segments by putting a border at the end of a term after which they perceived a shift in the message that the author intended to communicate, i.e., a different communication goal is pursued.*. For each segment, they were asked to provide a description (label) of 1-5 keywords. In order not to bias the annotators to look for specific segments, no limit on the size of a segment or the number of segments was specified nor were labels predefined. The task was performed online through a PHP, JavaScript application we developed for this purpose *and the outcome was 4.7K labeled segments.* The mean number of segments per post was found to be 4.2 for the HP Forum and 5.2 for the TripAdvisor.

Examining Human Segmentations

A. Verification of Segment Existence. The granularity of individual segmentations, as it was expected, varied. To verify that forum posts can be naturally divided into parts, we measured the annotation agreement. We considered *observed agreement percentage* (that shows how many annotators agreed over all) and *Fleiss's κ* that indicates whether the high observed agreement percentage is (or is not) due to chance agreement. We considered an offset from 10-40 characters (ref. Table 3.2). Within an offset of 40 characters (i.e., 3-5 terms with spaces and punctuation included), average observed agreement

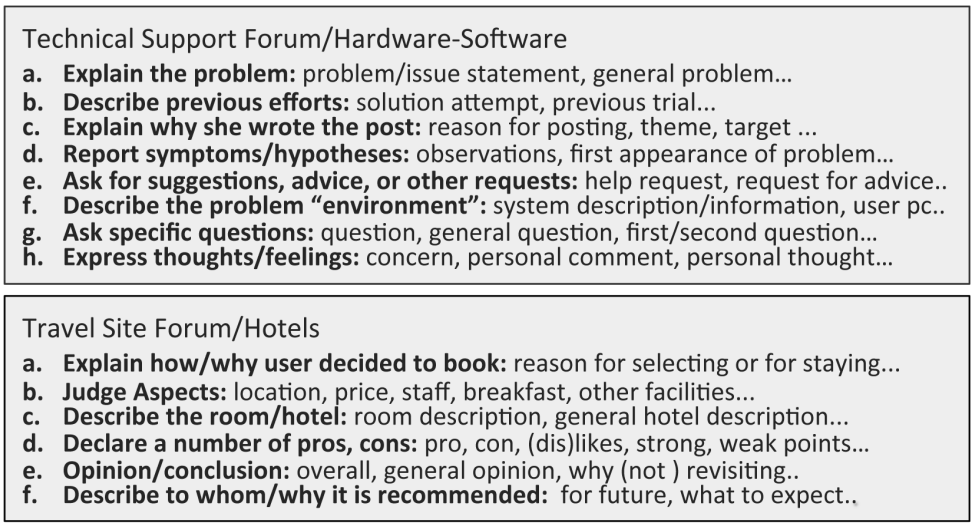


Figure 3.7: Annotators' labels, grouped in categories, for the goals that the segments are intended for.

percentage varies from 77% to 83%, where 0 indicates complete disagreement and 100% perfect agreement. Considering the strictest character offset (i.e., 10 chars, 1-2 terms) the agreement remains high (64% to 71%). Fleiss's κ, with negative values indicating lack of agreement and with positive values from slight to perfect agreement closer to 1, varies from 0.68 to 0.74. These values indicate *considerable agreement.* Thus, *posts are indeed organized into logical units that are relatively easily recognizable by humans.*

B. Underlying Messages. Since the labels of *the different messages that the authors communicate* were not predefined, there was a large variety of keywords describing the same message and that made the analysis of the results more complex. However, a predefined list of labels would have worked as a bias while for us it was important to crosscheck that the users would detect similar underlying author intentions with the ones we had observed without being directed to do so. The outcome was positive: labels such as *expectation*, *previous efforts*, *help request*, *hotel description* and *system description* were selected by the annotators. These labels do not describe what a segment actually talks about, which is what topics derived from LDA would have done, for instance, but indicate why the author wrote the specific segment. Segments with similar content (i.e., considering similar terms) have been labeled differently; while segments that do not share common terms have been labeled with the same or similar labels. Figure 3.7 *summarizes the most common labels* clustered into 7-8 categories for each dataset.

Automatic Segmentation Effectiveness

Given the posts from the two datasets that have been segmented by humans in the user study, we examined how much we can approximate human performance with different automatic segmentation approaches to: (*i*) evaluate the document representation based on CMs vs on a term-based one, (*ii*) select the most appropriate border selection mechanism, and (*iii*) evaluate the coherence/depth functions.

To measure how close an automatic segmentation is to human ones, we used a well-known metric from Computational Linguistics where originally segmentation task comes from the *multWinDiff*, a variation of the traditional *winDif* error which handles different number of annotations per post [Kazantseva and Szpakowicz, 2012].

Error. The *multWinDiff* error uses $|d| - m$ overlapping windows, where m is the size of window and equals to half average length of reference segmentations for the current document, and $|d|$ is the document size. It penalizes near-hits and misses based on how far the reference border and borders to-be-evaluated are. The penalization (or reward respectively) is stronger when more than one annotators have (or have not) placed a border at a certain point in the document.

$$multWinDiff = \frac{1}{|A|(|d|-m)} \sum_{a=1}^{|A|} \sum_{i=1}^{|d|-m} (|Oc_a \neq Oc_i|) \tag{3.10}$$

where A: all available annotations, $|A|$: their total number, $|d|$: the size of the document in text units, m: the window size, $|d| - m$: the number of windows and $|Oc_{ai} \neq Oc_i|$ is 0 when the number of borders of both segmentations is the same within this window, 1 otherwise.

A. Intention Representation: CM vs Term-based features. We tried out Hearst's segmentation algorithm that defines cohesive segments as *homogeneously lexically distributed text parts* and evaluates candidate borders using *cosine similarity on weighted terms*. We compared to our *Tile* strategy (ref. Section 3.6.3) that uses the same mechanism for border selection as the Hearst's segmentation algorithm but it represents documents as *vectors of their CM Features* (ref. Table 3.1); for the and cosine dissimilarity for the border score cosine dissimilarity was used. We observed that with *Tile*, the *average error is reduced by 18% (from 0.64 to 0.46), in the HP Forum dataset and by 26% in the TripAdvisor dataset*. The significant error reduction shows that CMs represent documents better when it comes to identifying borders that reflect shifts in intention.

B. Border Selection Mechanism Effectiveness. Subsequently, we performed a comparison of our border identification mechanisms, namely *Tile*, *Greedy* and *StepbyStep*. In all cases, we used the CMs described above and the score function of Equation 3.4, where

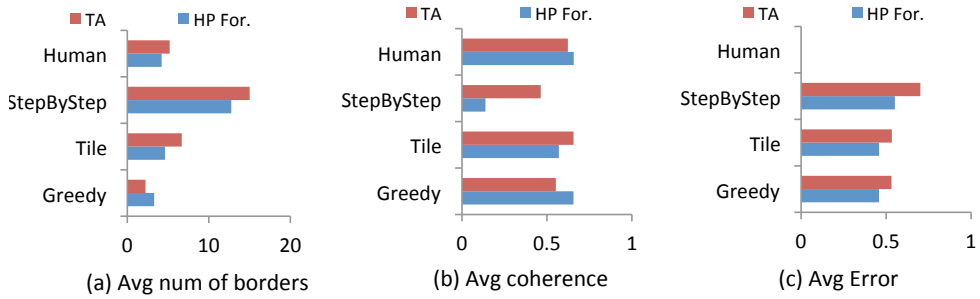


Figure 3.8: Comparison of border selection mechanisms

coherence is determined by Shannon's diversity and sentences as text units. Sentences are usually written to express a single complete message and they contain all (or almost all) communication means features. Thus, they constitute natural and intuitive text units. Fig. 3.8(a) shows the average number of borders. *Tile* returns more borders per post on average while *Greedy* less than human annotators for all the data samples. *StepbyStep*, on the other hand, returns way more borders. We observe that the first two mechanisms produce the most coherent segments after human segmentations (Figure. 3.8(b)) and have *the lowest error*, i.e., they approximate better human segmentations (Figure 3.8(c)). Thus, both *Greedy* and *Tile* look promising. We selected *Greedy* for the overall evaluation experiment.

C. Coherence and Depth Functions Comparison. We experimented with the Shannon's index and richness (for coherence) and with the distance functions: cosine dissimilarity, Euclidean distance, Manhattan distance (for depth). We found that Shannon's index diversity on CMs reduces the error the most: by 24%. In all the experiments we use CM representation and the reference point for our comparison is the multWinDif error of the topical segmentation (Hearst' algorithm). The table in Figure 3.9 summarizes the results. For a better understanding of the results, we provide, apart from the average error changes, the percentage of posts in which the segmentation of a post was approximated better, worse or the same, i.e., error reduction, increase and no change, respectively.

Function	*Posts with Error Decrease*	*Posts with No Change*	*Posts with Error Increase*	*Avg Error Decrease*
Cos.Sim.	68%	19%	11.5%	-0.18
Eucl.Dist.	64.7%	8.1%	29.83%	-0.22
Manh.Dist.	43.4%	10.7%	45.8%	-0.13
Richness	46.8%	11.5%	41.8%	-0.17
Shan.Div.	79.9%	15.5%	4.7%	-0.24

Figure 3.9: Error under different coherence/depth functions

3.9.2 Overall Technique Evaluation

Our approach on the recommendation of related posts has been evaluated in the environment of (*i*) a tech support forum by users of the forum trusted as *experts* (HP Forum dataset), and (*ii*) a well-known crowd-sourcing platform (CrowdFlower) by workers there (TripAdvisor dataset), and (*iii*) in the environment of a programming forum by computer scientists and engineers that regularly use the site (StackOverFlow).

The Methods. The choice of the methods to compare with was done in order to evaluate: (*i*) how matching posts at segment granularity compares with matching at the post-as-a-whole level; and (*ii*) the effectiveness of our segmentation and clustering processes. *For comparison with methods considering posts as a whole* we used the implementation for full-text matching included in MySQL 5.5.3. that uses the weighting scheme is described in Eq. 3.7 and a ranking method that is a variation of BM25[2]. This method will be denoted as *FullText*. We also used matching based on LDA topics with Gibbs sampling (denoted as *LDA*) [Blei, 2012; Blei et al., 2003].
For evaluating our segmentation and segment grouping processes, we examined how our matching method (ref. Algorithms 1 and 2) performs when the default segmentation into sentences is used instead of our border selection mechanism (based on intention shifts). We refer to this method as *SentIntent-MR*. We also considered our matching method using clusters of segments with similar content instead of intention clusters. Specifically, instead of the intention-based segmentation we performed a very well known segmentation based on topic shift [Hearst, '97] and clustering on TF/IDF vector representations of the posts. We refer to this method as *Content-MR*. Our proposed, complete, method is denoted as *IntentIntent-MR*. *MR* in the three last methods stands for *Multiple Ranking lists* and indicates the use of Algorithm 2; what changes is the type and content of clusters.

Subsection 3.9.3 provides more details about the different derived clusters. Subsequently, Subsection 3.9.2 compares the final top-k lists for different values of k, and data collection sizes.

Retrieved Top Lists

Considering as post-query every post in the HP Forum and TripAdvisor collection (i.e., 100k and 33k query-posts respectively), we run all the (five) methods. We next present how different are the derived lists of our method comparing to the other methods. Figure 3.10 illustrates an overlapping with FullText results: 11% and 6.2% respectively. This small overlapping is expected since, although the alternative methods make more often

[2]For a clear and fair comparison, the same ranking method (modified accordingly as described in Section 3.8 to consider intention clusters) was used for the comparison among segments in our method as well.

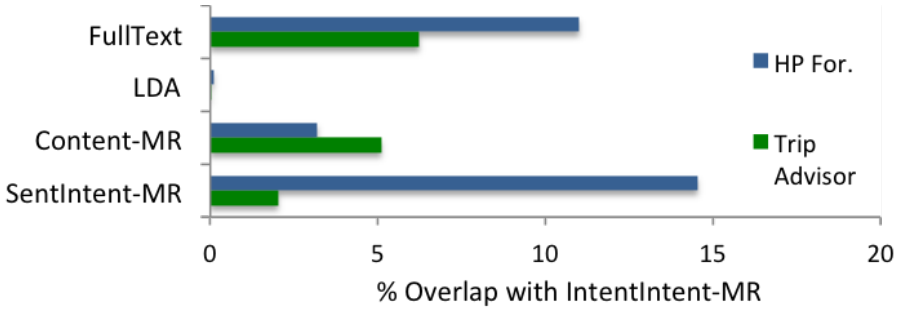


Figure 3.10: Overlapping of the posts in the top-5 lists retrieved by our method (IntentIntent-MR) and the rest of the methods considering the post-queries in the collections HP forum and Trip Advisor.

different choices, there are cases where content similarity is strong both in the documents as a whole and in the individual document parts within the segment clusters. The same applies for Content-MR as well where the respective overlapping is: 3.1% and 6.43% . It is worth noting that Content-MR and FullText return a lot of common results with an overlapping reaching 65%. Another interesting observation is that LDA returns completely different results. Without examining the quality of the results yet, we expect that LDA returns either significantly better or more erroneous results.

Additionally, we run our method and the baseline FullText for the StackOverFlow dataset considering as post-queries a sample of 6k post queries. It shows 10% overlapping in the top-5 lists, and 16.5% overlapping in the top-10 lists.

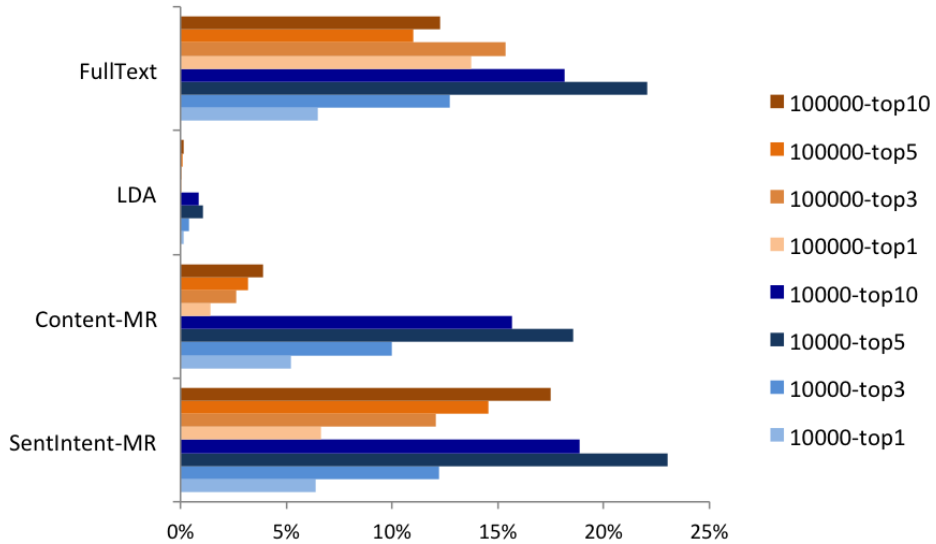


Figure 3.11: Overlapping with IntentIntent-MR in HP Forums dataset of 10k and 100k.

We also examined how different or similar are the results of each method when executed for *datasets of different size*. Specifically, we have considered apart from the product support forum collection of 100k posts, another smaller collection from the same dataset

consisting of 10k posts. and run all the (five) methods considering as post-query each post in the two collections. For the three *MR* methods, we first run their segmentation and grouping steps. For each document, the top 10 documents with the greatest matching scores are returned. Figure 3.11 illustrates the overlapping of the result lists of our method IntentIntent-MR with the lists returned by the rest of the methods. We present the overlapping in the first 1, 2, 3, 5 and 10 positions of the result lists to show whether the methods return the same results in the beginning or the end of their lists.

Both in the smallest and largest datasets respectively, the top-10 lists derived from SentIntent-MR exhibits the largest overlapping with our method, 18.9% and 17.47% respectively. Considering that less than 20% overlapping in the top-10 lists (and 23% in the top-5 lists) is not that significant overlapping, we understand that by removing the intention-shift identification and using sentences instead, a radical change occurs in the algorithm. The overlapping with FullText is close to SentIntent-MR (18.1%) in the top-10 lists of the small dataset but it reduces significantly to 12% in the larger dataset. A similar reduction is observed in the overlapping with Content-MR as well. Therefore, when there exist less documents to choose from (smaller datasets) the derived lists of the content-based and intention-based methods . Regarding Content-MR and FullText, we should also highlight that the overlapping in the top-5 lists is larger than the overlapping in top-3 and top-1 lists. Thus, we understand that although the other methods make more often different choices from ours regarding the first 2 positions of their lists, when there is strong content similarity along two documents, both our method and content-based methods select it. This happens at most in 22% of the lists in the small dataset and 15.38% of the lists in the large one which means that the result lists are disjoint in most of the cases.

User Evaluation

From each of the post collections described in the beginning of Section 3.9, we randomly selected some posts to serve as reference documents, i.e., d_q. The random selection gave us representative samples with segmentation granularity distribution very close to that of the whole datasets (ref. Table 3.3). For each of the sample document queries from the HP Forum and TripAdvisor datasets, users evaluated the top-5 posts returned by each method (ours and the alternative methods), while for the StackOverFlow dataset, users have evaluated the top-5 lists derived from our method and the best baseline (i.e., FullText). Every post-to-post matching, i.e., post pair, was evaluated by at least three users. We chose a binary evaluation over graded [Kekalainen, 2005] since we are interested in returning to the user only highly related posts. The derived dataset is described in Table 3.4. The five lists (one for each method) in the environment of the tech support

	HP Forum	
	All	**Sample**
1 Segs	30.7%	28%
2 Segs	40.5%	42%
3 Segs	28.4%	29.5%
4 Segs	0.37%	0.5%

	TripAdv.	
	All	**Sample**
1 Segs	25.1%	36%
2 Segs	46.1%	40%
3 Segs	23.5%	16%
4 Segs	4.8%	8%

	StackOverFlow	
	All	**Sample**
1 Segs	53.6%	30%
2 Segs	41%	50%
3 Segs	6%	20%
4 Segs	0%	0%

Table 3.3: Segmentation Granularity of Sample Post Queries

forum, and the two lists in the StackOverFlow were evaluated separately while for the TripAdvisor posts we performed pooling to generate a single list per query-post [Jones et al., 1975]. The user-experts evaluated the recommended forum posts in the lists having no information about how they had been generated.

	HP Forum (100K)	*TripAdvisor* (33K)	*StackOverFlow* (1.5M)
Methods	All	All	2
Post pairs	5000	750	240
Evaluations	15000	2193	1440
User Agreement	0.87	0.81	0.794

Table 3.4: Test Corpus

The inter-rater agreement (Fleiss' kappa) for the total was found to be: 0.87, 0.81, and 0.794 (for the HP Forum, TripAdvisor and StackOverFlow datasets, respectively) reflecting almost perfect agreement.

The evaluations were used to estimate the mean precision: the mean of the precision values considering each information need, i.e., post query, separately. Table 3.5 illustrates the results that are discussed in the next subsections.

Forum	*LDA*	*FullText*	*Content-MR*	*SentIntent-MR*	**IntentIntent-MR**	*Gain*$^{(*)}$
HP	0.01	0.16	0.065	0.16	**0.26**	**+10%**
TripAdvisor	0.21	0.53	0.27	0.45	**0.65**	**+12%**
StackOverFlow	-	0.161	-	-	**0.262**	**+10.1%**

(*)Considering the best baseline, i.e., FullText.

Table 3.5: Comparison of Methods - Mean Precision

Comparison with Baseline Methods

Full-text comparison matches to the post at hand d_q posts that share important, according to the used weighting scheme, common terms. Table 3.5 shows a clear *gain of 10%, 12%, and 10.1% in mean precision* for the three datasets, respectively. Our method, *IntentIntent-MR*, retrieves the most lists with the largest number of related posts in the first two datasets (ref. Fig. 3.13). Moreover, for the StackOverFlow dataset, *it reduces the lists with no true positives (mean precision 0) by 28.6% (ref. Fig. 3.12) while for the HP forum it is reduced by 24.5%.*

The higher precision is justified by the fact that common terms that appear in segments that are meant for a different purpose often lead to false positives. On the flip side, intention-based segmentation and grouping manages to distinguish the different messages before proceeding with the comparison step. Consequently, such false positives are avoided with *IntentIntent-MR*.

On the other hand, the *LDA method* performs worse than both our method and the *FullText* method. Specifically, Table 3.5 indicates *25% and 44% lower mean precision* than ours. We tried out topic-based comparisons as well since one could claim that they may exist terms correlated with different intentions that will allow such a comparison to distinguish the different intended messages without the need of segmentation. An over-simplified example would be the topics: "ink, blink, light, question" and "ink, blink, light, tried, unsuccessfully". Two documents that share the terms "ink, blink, light" would be not considered as related if they have been assigned to the two different topics describing a question and a user's effort respectively. However, we see that although topics describe posts at a higher level than that of terms, they fail to compare effectively posts that already belong to the same category.

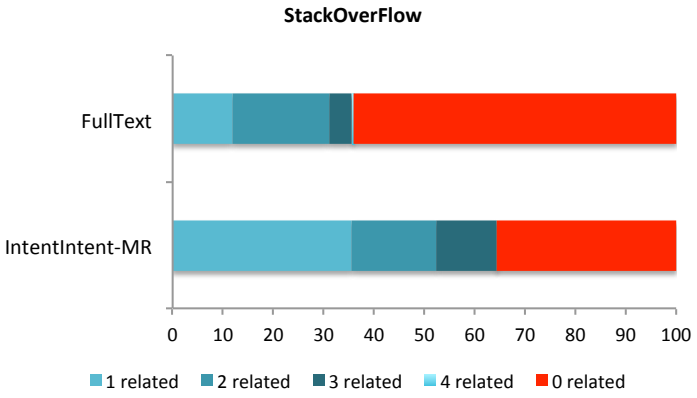


Figure 3.12: Retrieved lists by FullText and IntentIntent-MR.

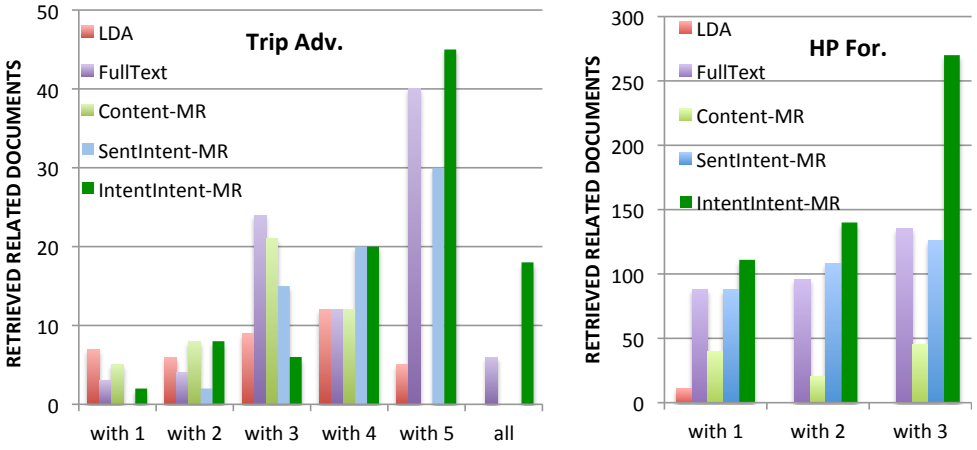


Figure 3.13: True Positives retrieved by the examined methods.

Comparison with Alternative Segmentation Methods

The comparison of our method with *Content-MR* (ref. Table 3.5) shows that forming clusters of segments that reflect different topics instead of intention clusters gives worse results (-19.5% and -39%). Consequently, term-based features can not effectively distinguish the different messages. In cases of collections with posts from different categories, *Content-MR* was found to perform better. However, the scope of our work is matching posts within the same Forum category by exploiting what the author intends to communicate; therefore, we do not get into these results.

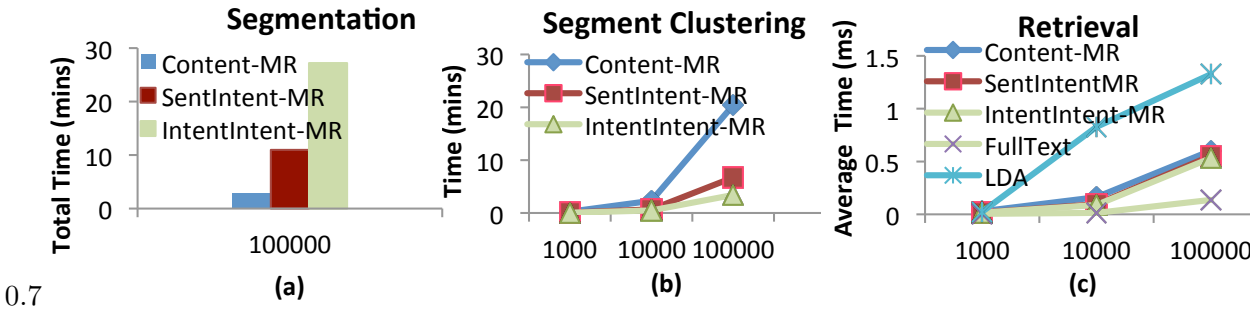
Moreover, *SentIntent-MR*, which creates *clusters of sentences* instead of clusters of segments based on the diversity of CM features (i.e., border selection step is omitted), shows performance closer to that of the *FullText* method that considers the posts as a whole and is lower than our complete method, *IntentIntent-MR*, by -10% and -20% (ref. Table 3.5). This comparison tells us that, without the border selection step, the segment grouping step fails to form the intention clusters degrading the performance of the matching algorithm. This verifies that the diversity in CMs manages to distinguish the different messages that the authors want to communicate.

Scaling

Avg Segmentation Time	*Total Segment Grouping Time*	*Avg Retrieval/Matching Time*
0.067 sec	3.18 min	0.029 sec

Table 3.6: Execution times (StackOverFlow dataset)

We have compared the time efficiency of our method to the other four methods considering the dataset of the product forum divided into three sets of 1k, 10k, and 100k posts,



0.7

Figure 3.14: Comparison of execution times (HP Forum Dataset)

respectively. Moreover, we have examined how our method behaves in a larger dataset, the StackOverflow.

Segmentation. Figure 3.14(a) illustrates the sum of the execution times of segmenting all the posts in the collection of 100k, i.e., the worst-case scenario where the post segmentation is performed sequentially. The segmentation is based on: intention shifts in *IntentIntent-MR* (greedy technique), topic shifts in *Content-MR*, and segmentation into sentences for *SentIntent-MR*. *IntentIntent-MR* requires about 60% more time than *SentIntent-MR* due to the additional border selection mechanism, while *Content-MR*, which requires no preprocessing (i.e., no POS-tagging etc) takes the less time. However, when the later segmentation method is used, the matching method manages to retrieve fewer true positives (ref. Table 3.5). The average segmentation time of our method for the product forum posts is 0.016 sec. On the other hand, for the second collection (StackOverFlow) (ref. Table 3.6), it is 0.067 sec. To run the segmentation, we first divided the dataset in 32 parts (1M lines per part) and run in parallel the segmentation of 5 to 7 parts. The execution time per part was 3.7h in average and the maximum 6.99h; while in total the segmentation of the 1.5M posts lasted 23 hours. All the reported times include html and special symbols cleaning, POS tagging and CM annotation; while for the second dataset there is an additional cost for reading the data in xml format, and selecting only the root posts with accepted answers.

Clustering or Segment Grouping is run on the whole dataset. Text clustering in general is computationally expensive. However, Figure 3.14(b) shows that in our case is efficient. The reason is that in the grouping step we represent text segments by only 28 numeric features (ref. Eq. 3.5, 3.6). The same applies for *SentIntent-MR*. The execution of the latter, however, lasts more since the number of sentences is larger than the number of segments. In all cases, clustering was performed using Weka 1.4 library. For the segment clustering of StackOverFlow dataset, we used a library that is intended for very large datasets and scales better [Achtert et al., 2013]. In fact it takes only about 3 mins for the 2.93M segments derived in the segmentation step (Table 3.6).

Matching, i.e., the top-k list retrieval given a post-query is also very efficient. Figure 3.14(c) shows that the average retrieval time in the product forum collection varies from 0.017 to 0.53 msec. The times of the methods that use multiple lists are very close. The fastest response time is that of *FullText* (less than 0.14 msec) because it accesses a single term index to get its answers. *LDA*, due to the lack of any indexing is the slowest (1.33 msec). Moreover, as Table 3.6 indicates, the average retrieval time in the StackOverFlow collection is only 2.9 msecs; i.e., less than 6 times higher response time for a 15 times bigger dataset.

3.9.3 Derived Segment Clusters

Our method (Intent-MR) produces 4 intention clusters for the HP dataset, 5 for the TripAdvisor, and 3 for the StackOverFlow dataset. On the other hand, Content-MR produces 10 and 8 clusters for the HP and TripAdvisor collection; while SentIntent-MR produces 3 and 5 clusters respectively. In the grouping step, the information about the assigned intentions is used to refine the borders that have been derived in the segmentation step, e.g., a document with three segments assigned to $\{I_2, I_0, I_2\}$ respectively will remain with two segments. Table 3.7 illustrates the granularity of the segmentation before and after the grouping step. In the end, the 30.7%, 25.1%, and 53.6% of the posts of the three datasets remain undivided, i.e., with only one segment. The remaining posts contain 2-4 different messages, while right after segmentation the granularity was between 1-8 segments for the first two datasets, and 1-4 for the last one.

	Before Grouping			After Grouping		
	HP	*Trip Advisor*	*Stack*	*HP Forum*	*Trip Advisor*	*Stack OverFl.*
1	25.1%	19.9%	43.3%	30.7%	25.1%	53.6%
2	25.1%	23.8%	30.6%	40.5%	46.1%	41%
3	18.8%	19.8%	14%	28.4%	23.5%	6.3%
4	16.36%	13.4%	6%	0.37%	4.8%	
5 – 8	39.6%	22.9%	0.55%			

Table 3.7: Segment Granularity - Percentage of Segments

Next, Figure 3.15 illustrates how many posts (specifically the percentage of posts) have segments in each of the final clusters. And in the sequel, we provide more details about the derived *intention clusters*.

To understand better the results, we concatenated the segments of each intention cluster into a single document (*intention-document*). For each dataset, we selected the top-100 most frequent terms in the *intention-documents* and in the corpus with the whole posts.

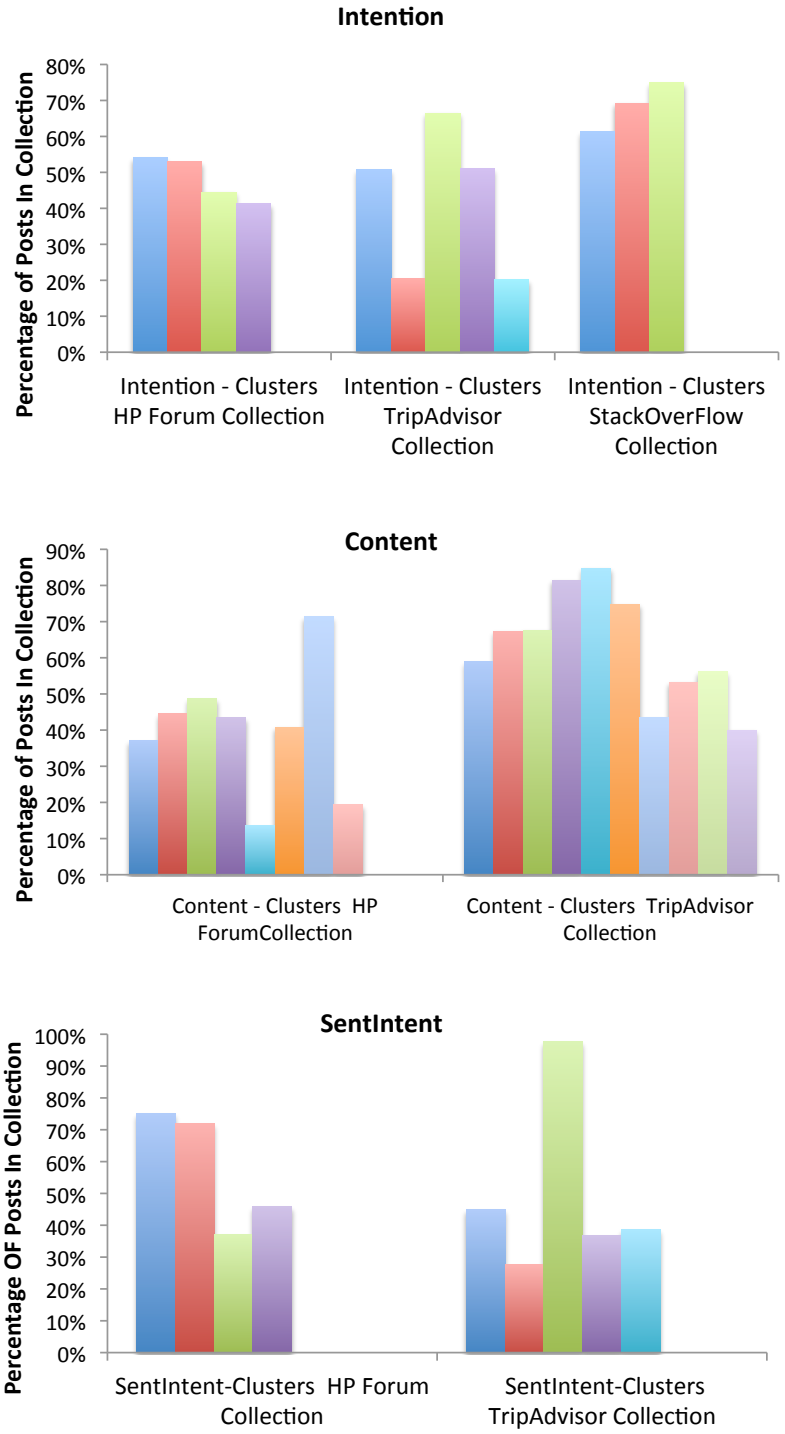


Figure 3.15: Distribution of posts (i.e., the respective segments) to the different derived clusters.

In Table 3.8, we present the terms that belong to the top terms of each intention document but not in the top terms of any of the other intention documents or the post corpus. There

HP Forum	
I_0	blue, status, recognize, question, feature, queue, press, tells, icon
I_1	advice, idea, solve, greatly, hope, utility
I_2	suddenly, successfully, moved, cleaned, plugged, alignment, previous, looked, removed, noticed, avail, needed, finally, unplugged, happened, weeks, told, shut, yesterday, wanted, suggested, received, made, lost, forum, thought, tech, gave, called, luck
I_3	[]
Trip Advisor	
I'_0	continental, beautiful, complimentary, fridge, standards
I'_1	person, impressed, recently, chose, wife, offered, moved, recommended, worried, included, manager, decided, cleaned, site, disappointed, pleased, surprised, pleasantly
I'_2	love, wouldnt, absolutely, travel, definitely, reasonable, star, friends, booking
I'_3	[]
I'_4	taxi, real, busy, middle, macys, west, hard, stop, airport, leave, prices, charge, cost, windows, floors, shops, cold, line, store
Stack OverFlow	
I''_0	[]
I''_1	height, document, length, open, title, import, action, item
I''_2	appreciated, missing, idea, understand, correct, android, figure, answer, point, added, inside

Table 3.8: Intention Clusters-*Distinguishing Top Terms*

exists one cluster in each dataset where no "not common" terms are found.

Our aim is not to label the intention-clusters. However, the sets of these terms give us a first idea of what is found in each intention cluster. Specifically, considering the labels assigned by the users in the segmentation user study (ref. Figure 3.7), I_0 can be intuitively matched to "explain the problem" or "ask specific question", I_1 to ask for advice or other requests, while I_2 to describe previous efforts. Moreover, I'_0 can be matched to "judge aspects", I'_1 to explain why the user decided to book, while I'_2 to overall opinion or conclusion and I'_4 to general hotel description. On the other hand, I''_2 can be matched to "ask a question", I''_1, contains most of the user efforts including code snippets, and I''_0 that contains all the top terms can be matched to a gerneral "problem description".

Cluster Evolution

Posts are dynamic data and as new data comes, it is natural that the intentions may

Old Cluster	New Cluster	Percentage of Old Cluster Items in the new
2015		
ini0-2015	new_0	99%
ini1-2015	new_1	51%
ini1-2015	new_2	48%
2014		
ini0-2014	new_2	100%
ini1-2014	new_1	92.3%
ini2-2014	new_0	89.77%

Table 3.9: Evolution of Intention Clusters in 2 years.

change and may need to be updated (i.e., the clusters should be recreated taking the new posts into account). The time efficiency of clustering (ref. Section 3.9.2) dictates that re-running the algorithm for the whole (updated) dataset is not a major issue that would require an incremental solution. We have also investigated the way that intentions change over time by performing a comparison between the intentions in the posts of two consecutive years from the StackOverFlow dataset and noticed no significant changes. We took the StackOverFlow data and we created 3 datasets that include: (i) the posts of 2014, (ii) the posts of 2015, and (iii) the concatenation of the two datasets (i.e., 2014 and 2015). And we have examined whether the segments of the posts (875410 segments in the first one and 731930 segments in the second dataset) that have been grouped together in the first two datasets, have been also placed together in the third case. In the dataset of 2015 have been formed 2 clusters, while in the one of 2014 3 clusters. When we merged the two datasets 3 clusters have been formed. What we have found is that the majority of the segments of each cluster from the two datasets have been also placed together in the clusters derived of the third dataset. Table 3.9 illustrates specifically that more than 89.7% of the segments of almost all the initial clusters (ini0-2015, ini0-2014, ini1-2014, ini2-2014) have been clustered together. Only one of the clusters in the first dataset ini1-2015 has been split.

Chapter 4

Recommendation through Goal Exploitation

In this Chapter, we present how we can deal with a very well-studied problem, the problem of recommendation, based on the principle of *goal-aware* selections, i.e., on the principle that human selections are not random and unrelated events but they are performed with the purpose of achieving some specific goal(s) that they have interest in fulfilling [Aarts and Hassin, 2005].

- In particular, we introduce and formally define the notion of goal-oriented recommendation, which evaluates every action considering the goals in which the current user may be willing to fulfill and how that action contributes to the fulfillment of one or more of these goals together with other actions from the user activity (Section 4.3).
- We explain how it differs from existing techniques and why the latter cannot be used to offer this type of recommendation (Section 4.2).
- We present different strategies for ranking the candidate actions, with each strategy implementing a different policy in prioritizing the goals and selecting the actions to be recommended (Section 4.4).
- We describe efficient ways of implementing the above strategies and materializing the goal oriented recommendation paradigm (Section 4.5).
- We perform an extensive experimental evaluation and we study the effectiveness of our methods and compare them to the state-of-the-art recommendation approaches proving that goal-based approaches can recommend actions that bring the user closer to the fulfillment of goals that are related to her/him, are highly different from each other and at the same time from actions performed by other users in the past (Section 4.7).

In what follows, the goal-oriented recommendation approach is introduced in Section 4.1, described in Section 4.3 and is placed in the related work in Section 4.2. The differ-

ent implementation strategies are introduced in Section 4.4 and their implementation in Section 4.5. The experimental evaluation and our findings are presented in Section 4.7.

4.1 Introduction

Recommender systems typically follow two approaches in their effort to help users to make their choices by identifying a small set of items from a collection that are most likely to be of interest to them: the collaborative and content-based filtering. The first approach is based on the idea that similar users have similar preferences, thus, successful recommendations to a user for unselected items can be made by analyzing the choices of similar users. On the other hand, the second approach, the content-based recommendation is based on the idea that users would like items that have similar features with items they have liked in the past. The principle behind both approaches is that whatever the past indicated as preference, it is likely to be preferred also in the future [Balakrishnan, 2010; Deshpande and Karypis, 2004; Li et al., 2004; Rendle et al., 2010; Sarwar et al., 2001]. However, although user selections may be of course affected by preferences, they are mainly results of specific goal(s) that a person has set and aims to fulfill [Aarts and Hassin, 2005].

We advocate that by recognizing the goals for which actions of the past have been performed, it is possible to identify the driving forces of the users' future actions and make recommendations that better fit these needs. Since the fulfillment of a specific goal may require actions that are highly different in nature, this form of recommendation may recommend actions that are highly different from those of the past, or from those that similar users have done in the past. Existing studies in recommender systems have already recognized that methods based on similarity with what has happened in the past are not always matching the user's expectations and have tried different techniques that focus on other aspects such as serendipity, novelty and diversity, in order to improve the recommendation quality. However, these solutions are not principled and are not driven by some specific user selected well-defined target. Note that we are focusing on the recommendation of "actions" and not "items", as typically done in recommender systems. Actions often refer to the *utilization* of items, e.g., a purchase of a product, or the watching of a movie but they may be more generic, e.g., *visit a doctor*, or *save up*.

Consider, for instance, the case of a customer in a supermarket that has placed in the cart a kilo of potatoes and carrots. A content-based recommendation will try to propose products that are close to what is already in her/his cart, i.e., similar to potatoes and carrots which means it may propose other kinds of vegetables, or even suggest other types of potatoes. On the other hand, a collaborative filtering system may suggest light beer or

red peppers, because these items have been bought in the past by customers with similar preferences. Both methods, through clearly different routes, recommend items based on the customer's past. Instead, by considering the fact that the items the customer has in the cart can be combined with other items to produce one or more food recipes, the system can open up new options to the customer. For instance, considering a recipe to make an olivier (russian) salad that includes: potatoes, carrots and pickles, an item to be recommended would be pickles. Another useful ingredient would be nutmeg that is a spice used for mashed potatoes and pan-fried carrots, two recipes that require products some of which are already in the customer's cart.

Such a recipe-based recommendation of products may not be justified by similarity to products already in the cart, neither by other product combinations found frequently in the carts of other customers. This means that neither association rules nor techniques that detect correlations among items can be employed to make such recommendations since they highly depend on the popularity these item sets have. So, unless we consider the product combinations found in the recipes, these products will not be recommended by other techniques. Furthermore, given the recipes, the recommendations can be optimized for an overall benefit. For example, products may be proposed that give the ability to the customers to maximize the recipes they can materialize.

Considering goals in the recommendation problem is challenging. The challenge comes from the fact that, in real life, there are typically multiple goals that one needs to fulfill at any given time. Each of these goals may require fewer or more actions in order to be fulfilled, and there may exist alternative ways for the fulfillment of a specific goal. Users have to reason on the priorities between the goals they try to achieve and the benefit they will have by the execution of each action towards the fulfillment of these goals. For instance, some users may prefer actions that help them fulfill a goal as soon as possible, while others may prefer actions that help the advancement of as many goals as possible. A goal-oriented recommender will have to leverage the goals by first recognizing the intended user goals, decide the priorities among them, and quantify the benefit of each action in relationship to the intended goals and in conjunction with the other possible actions.

We introduce a new family of recommendation strategies, i.e., *goal-based recommendations*, that deal with the above challenges. The goal-based strategies identify the goals for which exists evidence that the user is aiming at achieving. The evidence originates from the previous user activity, i.e., the actions that the user has already performed. Given this goal space, the strategies explore the sets of actions that lead to the fulfillment of these goals and contain actions that the user has already performed to find actions which the user has not performed and may be willing to complete. The sets of actions together with the goals they fulfill constitute the user's *goal implementation space*. The likelihood

that the users will like an action from the candidate set of actions in this space depends on their approach towards the goals they would like to fulfill. We have identified three different strategies for exploring and exploiting the user's spaces in order to select the actions to be recommended. The three strategies correspond to three different policies based on which users often make their selections.

The first strategy is the *Focus* that examines each of the action sets in the user's goal implementation set to find which of them lead to the fulfillment of the goal that is closest to completion, either because most of the required actions have been already performed ($Focus_{cmp}$), or because they require only a few more actions ($Focus_{cl}$). Then, it forms the recommendation lists from the actions in these action sets. It is the policy preferred by users that need to fulfill at least one goal through the actions in the current recommendation list. The second strategy, *Breadth*, is not examining each action set in the user's goal implementation space separately. It considers more than one set of actions at the same time. Specifically, it evaluates and ranks the actions in the user's action space based on all the sets this action participates and selects those actions that belong in as many sets as possible together with as many as possible actions from the user activity. This strategy is for users that would like to fulfill as many goals as possible, if possible, through this recommendation list, but in order to maximize the number of fulfilled goals, they are willing to complete some or all of them in the future, i.e., not only through the actions in the current recommendation list. This way it keeps some "paths" open for the future (i.e., unfulfilled goals) but those paths exploit the actions that the user has done so far. We also suggest a third strategy, the *Best Match*, that similarly to *Breadth* is not trying to fulfill at least one goal through the current recommendation list. It recommends actions that contribute to the goals of the user's goal space. However, in contrast to *Breadth*, *Best Match* evaluates an action considering the whole goal space, not only the goals to which this specific action contributes. It generates a profile for the user and estimates a similarity between this profile and the actions to be recommended. The action representation shows how much that action contributes to the fulfillment of the various goals and the user profile how many of the user actions contribute to the various goals. It is a policy that may end up in the fulfillment of many goals in the future. However, it is a strategy for users that are interested in actions that are more useful (contribute more) to the goals to which the user has has put more effort in the past (and respectively less to goals to which the user has put less effort).

4.2 Recommender Systems

[**Collaborative Filtering**] [Balakrishnan, 2010; Deshpande and Karypis, 2004; Li et al., 2004; Rendle et al., 2010] is one of the traditional methods that exploits previous item selections or interactions that similar users have performed. The principle of similar users showing similar preferences has been also adopted by group recommenders (recommenders that do not consider users individually but as groups) [Amer-Yahia et al., 2009; Ntoutsi et al., 2012; Stefanidis et al., 2012] Instead, our method does not look other users but the actions in the implementations of the various goals. It neither selects implementations similar to the user's past activity, but those that contain subsets of it and can be adequately extended to lead to the fulfillment of one or more goals. Thus, what we propose differs from what the user or other similar users have already done.

[**Content-based Filtering**] Another technique is the *Content-based Filtering* that employs similarity measures based on item characteristics to identify and recommend items similar to what the user has used in the past with a high degree of satisfaction. The set of characteristics can be enhanced with auxiliary data, such as, ontological classes [Middleton et al., 2009], knowledge graphs and other contextual factors [Adomavicius and Tuzhilin, 2008; Fouss et al., 2007]. Our method is not intended to retrieve items similar to the user's previous choices, thus, is different from the content-based filtering. Only one of the different policies we propose considers user-preferences but the preferences reflect the contribution of the user's actions to certain goals (ref. Section 4.4.3).

[**Association rule mining**] It analyzes the user's histories to identify groups of items appearing together and use this as the basis for making recommendation [Sandvig et al., 2007]. The approach is based on popularity, while our technique is not affected by popularity fluctuations. Furthermore, different actions may often appear together but for different goals, which means not only that recommendations different than ours will be made, but these recommendations may also be incomplete, i.e., manage to fulfill none of the goals, since the system is confused and unable to distinguish the different intended goals of the actions that appear together.

[**Next Action Prediction**] There is a family of works that try to predict the next action in a sequence, e.g., the next web page to click or the nest location to be [Keren et al., 2015; Sadikov et al., 2010]. The purpose of these systems is to guess the next action and not to recommend a set of actions of interest. Often their purpose is to promote the inferred actions or act in anticipation of the user's actions [Armentano and Amandi, 2009]. To infer the next action(s) they employ models such as probabilistic (state transition) models, e.g., Bayesian Networks [Patterson et al., 2003], or Markov models [Sadikov et al., 2010]

or other variations [Armentano and Amandi, 2009]. Such methods consider only the sequence of the latest actions to make the best guess. Furthermore, they work under the assumption that in any given point there is only one goal. So in some sense they are a specialized case of our problem, but their solution cannot be applied to ours. For instance, when our method generates recommendations considering a set of food recipes as the goal implementation set (ref. Section 4.7), it does not do next recipe or ingredient recommendation in the way of strict menu planning that recommenders do [Forbes and Zhu, 2011; Teng et al., 2012]. Instead it discovers interesting suggestions of the form *buy + "some product"* based on the recipes that are related to the user activity.

4.3 Goal-Based Recommendation Approach

[**Actions, Goals and Goal Implementations**] We assume the existence of a set $\mathcal{U}$ of *users*. Users perform tasks such as the purchase of an item, the visit of a web page, the watching of a movie, or any other recordable task. To model the tasks we consider the existence of a set $\mathcal{A}$ of *actions*.

People set the goals that they need to achieve and then they decide to perform those actions that they believe will help them fulfill their goals. We denote $\mathcal{G}$ the set of goals. A set of actions constitutes an *activity*, which means that there are $2^{\mathcal{A}}$ different possible activities. The activities that are intended for a goal $g \in \mathcal{G}$, alongside the respective goal, are referred to as *goal implementations*.

Definition 9 *A* goal implementation*, or simply implementation, is a pair* $\langle g, A \rangle$ *with* $A \in 2^{\mathcal{A}}$ *and* $g \in \mathcal{G}$*.*

Goal implementations can be found in sources related to almost every aspect of human activity. Recipes, for instance, are actually implementations of specific goals (the food that the recipe is about). Online learning platforms have specializations and degrees, that serve a purpose similar to goals. Each specialization is associated to one or more set of courses indicating the actions required to achieve the goal, i.e., the specialization. Goals can be found even in online stores. Many online clothing stores, for instance, give users the ability to form *outfits* and annotate them with labels such as 'for friend meetings", "to be warm" and so forth. Those outfits that enjoy high popularity may be considered as implementations of the goal, with the goal being the label. With this knowledge, when the system recommends some item to a user apart from considering the user preferences on characteristics such as color or material (content based filtering) or considering what clothes others have bought in the past (collaborative filtering), it can employ a goal-based

recommendation technique and suggest items that can be combined with clothes the user has bought in the past to form complete outfits.

Another rich source of goal implementations are the social networks or specialized web sites where users record and share success stories of things that they do in life. Examples include 43things[1] and WikiHow[2], where users describe actions to achieve real-life goals. There are many works on transforming such textual descriptions into a structured form, like an ontology [Jung et al., 2010; Pareti et al., 2014; Ryu et al., 2010], or a taxonomy [Chulef et al., 2001; Strohmaier et al., 2009]. They typically employ structural information such as HTML tags or enumeration. A different way to create such datasets from web pages is by posing queries of the form "in order to + *a goal description*" on search engines [Strohmaier et al., 2009].

One of the datasets that we are using in the experimental evaluation of this work contains 18k goal implementations that we have extracted by performing *action identification* on user-generated descriptions about everyday goals such as *learn english*, *travel to Italy* and so forth from the 43Things website. We did this action extraction with a module that we have developed for this purpose, that works on a simpler model and for plain text (ref. Section 4.5). We do not elaborate further on the extraction task here; it will be fully presented in Chapter 5.

Example 9 *Figure 4.1 depicts a set of goal implementations from an online clothing store. We denote a goal implementation space as L. The columns indicate outfit purposes (the goals) while the rows are the items (the actions). If we depict by a_k the action of buying the item i_k, then the implementation set is:*

Implementation	$\langle$*Goal,Activity*$\rangle$	
$p1$	$\langle g_1, A_1\rangle$	*where* $A_1=\{a_1, a_2, a_3\}$
$p2$	$\langle g_1, A_2\rangle$	*where* $A_2=\{a_1, a_2, a_4\}$
$p3$	$\langle g_2, A_3\rangle$	*where* $A_3=\{a_1, a_4, a_5\}$
$p4$	$\langle g_3, A_4\rangle$	*where* $A_4=\{a_3, a_5, a_6\}$
$p5$	$\langle g_3, A_5\rangle$	*where* $A_5=\{a_1, a_3, a_5, a_6\}$

An action a is said to *contribute* to a goal g through a goal implementation $p=\langle g, A\rangle$, denoted as $a \rightsquigarrow^p g$, if $a \in A$. It is possible that an action contributes to more than one goals. All these goals form the goal space of the action.

Definition 10 *Given a goal implementation set L, the* goal space *of an action a is the set $\mathcal{GS}(a)=\{g \mid g\in\mathcal{G} \wedge p\in L \wedge a\rightsquigarrow^p g\}$.*

[1] https://en.wikipedia.org/wiki/43_Things
[2] www.wikihow.com

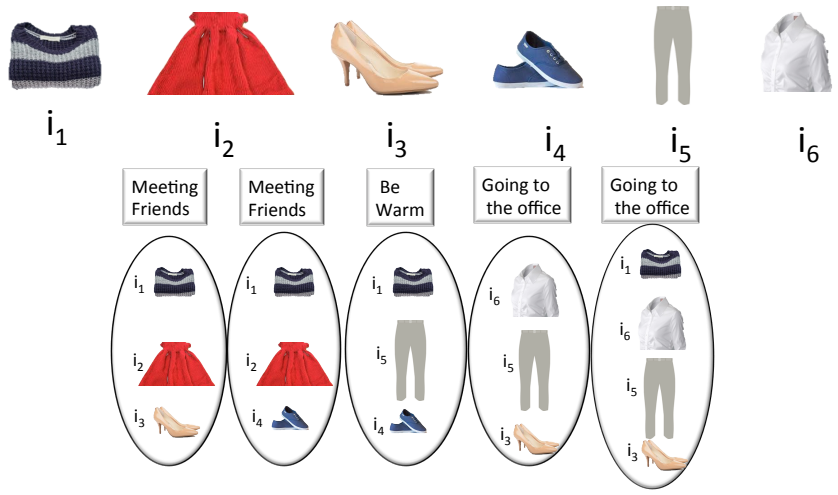


Figure 4.1: Combinations and the goal they serve

The implementations that contain a certain action to their activity form what is called its implementation space. The cardinality of the implementation space of an action may be larger than that of its goal space since it is possible that for the same goal there are more than one implementations involving the action.

Definition 11 *Given a goal implementation set L, the* implementation space *of an action a is the set $\mathcal{IS}(a)=\{p \mid g\in\mathcal{G} \wedge p\in L \wedge a\rightsquigarrow^p g\}$.*

Given an action, it is of interest to know what other actions need to be performed together with this action so as the goals in the goal space of that action to be fulfilled. The set of these actions forms the action space of an action.

Definition 12 *Given a goal implementation set L, the* action space *of an action a is the set $\mathcal{AS}(a)=\{a' \mid g\in\mathcal{G} \wedge p\in L \wedge a\rightsquigarrow^p g \wedge a'\rightsquigarrow^p g \wedge a\neq a'\}$.*

The goal space definition extends naturally to the case in which we have a set of actions A instead of one, to be the union of the goal spaces of the individual actions in A. In other words, $\mathcal{GS}(A) = \cup_{a\in A}\mathcal{GS}(a)$. The same extension applies also for the for the implementation space, i.e., $\mathcal{IS}(A)=\cup_{a\in A}\mathcal{IS}(a)$.

Example 10 *In the implementation set of Example 9,*
since action a_1 participates in the activities A_1, A_2, A_3 and A_5, its implementation space is the $\mathcal{IS}(a_1)=\{p_1, p_2, p_3, p_5\}$, and its goal space the $\mathcal{GS}(a_1)= \{g_1, g_2, g_3, g_5\}$. Its action space is the set of all the other actions in A_1, A_2, A_3 and A_5, i.e., $\mathcal{AS}(a_1)=\{a_2, a_3, a_4, a_5, a_6\}$.

Recommendation Setting We assume an implementation set L. The set may have been constructed through one of the many methods mentioned earlier that are already available in the literature, or through the text-based module we developed for the experimental evaluation.

We also assume that a user has performed a number of actions already. We refer to these actions that the user has performed as the *user activity* H. We do not know why these actions have been performed but given the goal implementation set, there is a number of possible goals that the user may have had in mind when performing each of these actions. These goals are actually the goal space of the activity H.

The goal space makes all the actions in the action space of the activity H to be likely of interest to the user since they will help towards the fulfilment of one or more of the goals in the goal space. Our aim is to recommend to the user actions that are not in H, and which the user would be happy to perform. However, not all the actions offer the same benefit. What action the user would be happier to perform depends on what priorities the user puts on the goals. Some actions may help towards the fulfillment of many goals, while others towards the fulfillment of goals almost completed. Thus, we need to create a ranked list of the actions to recommend according to some criterion. Depending on the criterion/policy we use, a different recommendation strategy is materialized. These policies comprise the topic of the next section.

4.4 Strategies

We have identified different options with which we believe users prioritize actions. These options correspond to two strategies: *Focus* and *Breadth*, described in the Subsections 4.4.1 and 4.4.2, respectively. Furthermore, we consider a methodology that in order to select actions builds a user profile that shows how the actions that the user has performed contribute to the various goals. Based on the user profile, it selects actions that contribute to the goals in the user's goal space similarly to the actions that the user has already performed.

4.4.1 Focus

We advocate that it is important to give to the user the option to have access to actions that lead to the *completion* of one of the goals in the user's goal space. Based on this idea, we introduce a strategy, referred to as *Focus*, that forms a recommendation list making sure that the actions in the list together with a subset of the user activity H form the

activity of a goal implementation in the library L, i.e., they comprise the actions that are required for the goal to be completed.

Algorithm 3 Focus Ranking

Ranks goal implementations based on completeness (step 3) or alternatively closeness (step 3)*

Returns the top k candidate actions from the top implementations

Input Set H, Set CA, int k

Output List R

1 $CI \leftarrow []$,$R \leftarrow []$
2 **for each** pId in $\mathcal{IS}(H)$
3 $\langle p, sc\rangle \leftarrow \langle\ pId,\ \frac{|A \cap H|}{|A|}\ \rangle$
3* $\langle p, sc\rangle \leftarrow \langle\ pId,\ \frac{1}{|A - H|}\ \rangle$
4 CI.add($\langle g, A\rangle$,$sc\rangle$)
5 **end for all**
6 rankCI based on sc
7 **while** CI and R.length<k
8 $CI_{curr} \leftarrow CI$.getNext()
9 $\langle g, A\rangle \leftarrow$GI-AV-idx[$CI_{curr}$.pId]
10 **while** R.length<k and A.hasNext()
11 $aId \rightarrow$A.getNext()
12 if $aId \in CA$
13 R.add($\langle\ aId,\ CI_{curr}$.sc$\rangle$)
14 **end while**
15 **end while** 16 return R /*it is already ranked*/

The question is which goals will be promoted in the list. For this purpose, our recommendation strategy first ranks the implementations of the goals in the user goal space, and then it recommends the actions that operationalize the highly ranked goal implementations (and by extension the highly ranked goals).

Given the user activity H, we consider the goal implementation space $\mathcal{IS}(H)$, i.e., the implementations of the goals in the user's goal space. Every implementation in $\mathcal{IS}(H)$ contains at least one action from the user activity and actions from the set of candidate actions. To rank the implementations (and hence the goals), we can follow two strategies.
− *Maximum Goal Implementation Completeness* ranks higher the goal implementations whose completed part (the actions that have been already performed and hence are part of H) is larger than their incomplete part (the remaining actions that have not been performed yet).
− *Maximum Goal Implementation Closeness* ranks higher those goal implementations that require fewer actions to be completed, i.e., they are *closer* to fulfillment.

Both strategies aim at the fulfillment of the selected goals, the difference is that the first one considers the most justified choice, i.e., the goal that the user has committed more to; while the second one promotes the fastest goal towards completion.

Maximum Goal Implementation Completeness. We measure the *completeness* of a goal implementation $\langle g, A\rangle$ by the fraction of the actions in the activity A of the implementation that at the same time belong to the user activity H.

$$completeness(\langle g, A\rangle, H) = |A \cap H|/|A|\rangle \tag{4.1}$$

We rank the goal implementations in $\mathcal{IS}(H)$ in descending order of completeness. Given this ranking, the list of action recommendations is formed as follows. We pull from the first goal implementation all the actions that have not been performed yet, i.e., that are not in the user activity, and we add them to the recommendation list. If more actions are needed for the top-k list, we pull the next goal implementation and so forth, until the list gets full. Note that it may be the case that the remaining slots in the top-k list are fewer than the actions of the next goal implementation in the ranked list of implementations. In this case, we can decide to leave the list with fewer recommendations or expand k to include the required actions for this implementation.

The completeness ranking function promotes the actions in the activities of the goal implementations with the largest completeness (see Algorithm *Focus Ranking*, line 3). This way the recommendation mechanism guides the user to actions that will lead to the fulfillment of the goal for which the user has already done most of the work, i.e., she has performed most of the actions needed for its fulfillment.

Maximum Goal Implementation Completeness is inspired by plan inference in plan libraries for intelligent agents [Carberry, 2001b]. However, in intelligent agents the aim is to predict which plan the agent is following (i.e., the agent has already selected a plan) while in our problem the recommendation mechanism aims to guide the user to options that s/he may have not considered without the recommendation system.

Maximum Goal Implementation Closeness. In contrast to the previous strategy, this strategy, i.e., *Maximum Goal Implementation Closeness*, considers only the remaining actions. It guides the user to follow implementations that will lead to the fulfillment of the goal that is *closest to fulfillment*. *The number of remaining actions*, i.e., the actions that have not been performed in each goal implementation of the space $\mathcal{IS}(H)$ indicates how close an implementation is to its completeness, i.e., its *closeness*. The goal implementation with the maximum closeness is the one that will be completed with fewer actions. Hence, we define the closeness of a goal implementation as the inverse number of remaining actions.

$$closeness(\langle g, A\rangle, H) = 1/|A - H| \tag{4.2}$$

We rank the goal implementations in $\mathcal{IS}(H)$ in descending order of closeness. Given this ranking, the list of action recommendations is formed as explained earlier (for the maximum completeness ranking). Closeness constitutes the score *sc* in the Algorithm *Focus Ranking* (line 3* is considered instead of line 3).

4.4.2 Breadth

With the strategy Focus, a single goal drives the recommendation process. This can be very restrictive if the user is not that determined to fulfill the specific goal. Therefore, we give the user another option: *Breadth* that evaluates every candidate action a considering a subset of goals in the goal space. This subset consists of the goals that are connected to the candidate action a through one or more goal implementations, i.e., the goal space $\mathcal{GS}(a)$. The reasoning behind this is that since every action in the action set $\mathcal{A}$ can participate at the same time in more than one goal implementations in the set L and possibly contribute to a number of goals, its benefit should be estimated based on all these goals.

First, in order to evaluate a candidate action a, we should take into consideration the number of goal implementations in its implementation space, i.e., the $\mathcal{IS}(a)$. We will refer to this quantity as *utility*. For an action a, its utility $u(a)$ is:

$$u(a) = |\mathcal{IS}(a)| \tag{4.3}$$

The larger the utility of an action, the larger the benefit that the user can have by a single action. For instance, in the Example 9, the action of buying item i_5 (i.e., a_5) that is part of three goal implementations: p_3, p_4, p_5 (i.e., it can be used in 3 different outfits) can be considered as more beneficial to the user compared to the action of buying i_6 (i.e., a_6) that contributes only through 2 goal implementations: p_4 and p_5. However, considering the user activity H=$\{a_2, a_3\}$, we remark that the user has not showed interest to goal implementation p_3 ($p_3 \notin \mathcal{IS}(H)$). Consequently, goal implementation p_3 should not have been taken into consideration. Thus, we need a measure that captures the utility of an action considering the user activity as well. For this purpose, we have *modified* Eq. 4.3 to consider a certain activity A. The implementation space $\mathcal{IS}(A)$ of the activity A determines the utility of the action a as follows:

$$u(a, A) = |\mathcal{IS}(a) \cap \mathcal{IS}(A)| \tag{4.4}$$

Therefore, considering the user activity H, the utility of an item a will be $u(a, H) = |\mathcal{IS}(a) \cap \mathcal{IS}(H)|$, which is in fact the number of goal implementations $\langle g, A\rangle$ in $\mathcal{IS}(H)$ where $\{a\}\subseteq A$.

Algorithm 4 Breadth Ranking

Ranks Candidate actions based on all the Implementations of the user's Implementation space where they participate

Input: Set H (user activity), int k

Output: List R

```
1   R ←[], CA ←AS(H)
2   actionScores← {}
3   for each pId in IS(H):
4      ActionsInP ← GI-AV-idx[pId]
5      ActionsInP←ActionsInP∩ H
6      for each aId in CA
7         if aId in actionScores.keys:
8            actionScores[a]← actionScores[a] +ActionsInP∩H
9         else:
10           actionScores.[aId]← 1.0
11  for each aID in CA
12     sc value is stored in actionScores[aId]
13     R.add(⟨ aId, actionScores[aId] ⟩)
14  rank R on sc and return top k actions
```

However, recommending actions of high utility is not enough. We should also consider how related, or else strongly connected, is a candidate action to the user activity. To do so, we need to consider how many of the actions in the user activity are connected to one of the goal implementations in the subspace $\mathcal{IS}(a) \cap \mathcal{IS}(H)$.

$$sc(a, H, Breadth) = \sum_{\forall \langle g,A \rangle \in \mathcal{IS}(a) \cap \mathcal{IS}(H)} \; \sum_{\forall a' \in A, if\ a'=a\ or\ a' \in H} 1 \tag{4.5}$$

The above equation captures both the utility of a candidate action and its relatedness to the user activity. Now, we can rank the candidate actions and get the recommendation list R. To form the recommendation list, we rank the candidate actions using the function described in Equation 4.5.

Algorithm *Breadth Ranking* presents in pseudocode the steps of the *Breadth*. The algorithm does not estimate the score (ref. Eq. 4.5) of each action in the $\mathcal{AS}(\mathcal{H})$ separately. It examines each implementation of the $\mathcal{IS}(\mathcal{H})$ and updates the score of all the actions of the $\mathcal{AS}(\mathcal{H})$ that belong in the current implementation. This way, when all the implementations have been examined the action scores (ref. Eq. 4.5) are ready and the action ranking takes place.

4.4.3 Best Match

We introduce the *Best Match* policy that in contrast to *Breadth* that evaluates each action in the user's goal space considering only the goals in the goal space to which this specific action contributes, it considers the whole goal space. In fact, it generates a user profile that reflects the effort that the user has made towards the goals and retrieves actions that contribute similarly to those goals. *Best Match* considering the user's goal space, represents every action as a vector and aggregates the representations of the individual actions in the user activity into a single vector. The final vector constitutes the user profile. Subsequently, the candidate actions can be ranked based on their similarity to the user profile. Such an approach promotes actions that contribute to most of the goals in the user's goal space.

Goal-based user representation. In recommendation systems, user profiles are described in terms of the features of the items that a user prefers. In our case, a profile captures the user dedication towards a set of goals. We consider that the more actions from the user activity H contribute to a specific goal in the goal space $\mathcal{GS}(\mathcal{H})$ of the user activity, the more the user cares for this goal.

Hence, we consider that an action a can be represented as a vector $\vec{a}$ in the feature space $F^{\mathcal{GS}(H)}$ (as an item is represented by considering features in content-based recommendation methods). One option would be to form a boolean vector, $\forall i \in \{0, |\mathcal{GS}(H)|\}$, where $g \leftarrow F^{\mathcal{GS}(H)}[i]$:

$$\vec{a}[i] = \begin{cases} 1, if \ \ \exists p \leftarrow \langle g, A \rangle \ \ s.t. \ \ a \in A, g \in \mathcal{GS}(\mathcal{H}) \\ 0, otherwise \end{cases} \tag{4.6}$$

The problem with the above representation is that it disregards the fact that an action in the user activity may contribute to a goal through one or more implementations. Therefore, instead of the boolean representation, we adopt a vector representation where $\vec{a}[i]$ is defined to be the number of goal implementations p s.t. $a \rightsquigarrow^{p} g$ and $g \in \mathcal{GS}(H)$. The value in each position of the vector $\vec{a}$ becomes $\forall i \in \{0, |\mathcal{GS}(H)|\}$, where $g \leftarrow F^{\mathcal{GS}(H)}[i]$:

$$\vec{a}[i] = \sum_{\forall p \leftarrow \langle g, A \rangle \ s.t. \ a \in A, \ g \in \mathcal{GS}(\mathcal{H})} 1, \tag{4.7}$$

To get the *user profile*, we aggregate all the representations of the actions of the user activity in the feature space $F^{\mathcal{GS}(\mathcal{H})}$ into a single vector. The user profile captures for each goal in $\mathcal{GS}(H)$ how many of the user actions contribute to this goal considering the different goal implementations for the same goal as well. Since the user profile is generated based on the current user activity H, we denote it as $\vec{H}$.

$$\vec{H} = \sum_{\forall a \in H} \vec{a} \tag{4.8}$$

Algorithm 5 Get-Goal-Based-Profile

Creates the user profile that reflects her connections with the Goal Space

Input: Set H (user activity)

Output: $\vec{H}$ vector in $\mathcal{GS}(H)$ that aggregates the contribution of all actions in H

```
1  H⃗←∅
2  GP_map←∅
3  for each aId in H :
4    IS ← IS(aId)
5    for each pId in IS:
6      gIdTmp ← GI-G-idx[pId]
7      if gIdTmp in GP_map.keys:
8        GP_map[gIdTmp]← GP_map[gIdTmp]+1
9      else:
10       GP_map[gIdTmp]←1
11  /*convert map GP_map to a vector in F^GS(H) space*/
12  for each gId in GS(H)
13    H⃗.add(GP_map[gId])
```

For example, for the user activity: H=$\{a_2, a_3\}$, the number of goal implementations where at least one of the actions of the user activity participate is 4. The user profile is $\vec{H}$ =$\{$ 3, 0, 2 $\}$. In the user profile is reflected the fact that the user has performed a_1 and a_3 that contribute to g_1: "*meeting friends*" 3 times and to g_3:"*going to the office*" via one goal implementations each, and that the user has shown her/his preference to the goals g_1 and g_2 over the rest of the goals in the goal space $\mathcal{GS}(H)$.

Goal-based representation of candidate actions. To rank the candidate actions against the user profile, we represent each candidate action in the same goal space, i.e., as goal vectors in the space $F^{\mathcal{GS}(\mathcal{H})}$ in the exact same way the actions from the user activity have been represented (ref. Eq. 4.7).

Distance-based Ranking. To rank the candidate actions, we can use a standard similarity or distance metric such as Euclidean distance between the user profile and each of the candidate actions, as follows:

$$sc(a, H, Best\ Match) = dist(\vec{H}, \vec{a}) \tag{4.9}$$

For instance, considering the Example 9, action a_1 from the user activity H would be closer (smaller distance) to the user profile than that of a_4 since the first contributes to g_1: "*meeting friends*" via two goal implementations and via another goal implementation to g_3:"*going to the office*" as well; while the latter contributes to g_1 via only one goal implementation and to g_2:"be warm" to which the user has shown no interest.

Algorithm 6 Best Match Ranking

Ranks actions based on their distance to the goal-based user profile

Input user activity H, int k

Output List R

```
1  R←[], CA←IS(/A)
2  H⃗←Get-GoalBased-Profile(H)
3  for each aId in CA:
4    GP_map←∅
5    IS ← IS(aId)
6    for each pId in IS:
7      gIdTmp ← GI-G-idx[pId]
8      if gIdTmp in GP_map.keys:
9        GP_map[gIdTmp]← GP_map[gIdTmp]+1
10     else:
11       GP_map[gIdTmp]←1
12   /*convert map GP_map to a vector in F^GS(H) space*/
13   for each gId in GS(H)
14     a⃗.add(GP_map[gId])
15   ⟨aId, sc⟩←⟨ aId, dist(a⃗,H⃗)⟩
16   R.add(⟨ aId, sc ⟩)
17 Rank R on sc and return top k actions
```

Algorithms *Get-GoalBased-Profile* and *Best Match Ranking* describe the procedure. *Get-GoalBased-Profile* forms the goal-based vector representation of the user (user profile) by considering for each action in the user's activity all the implementations where the examined action belongs (i.e., its implementation space) in order to find to which of the goals of the user's goal space it contributes and add one in the respective position of the vector $\vec{H}$. On the other hand, *Best Match Ranking* compares the user profile with the goal-based vector representation of each action in $\mathcal{CA}$ by considering again the goal implementation space of the actions and the goals to which they contribute. and finally ranks them according to their distance with the user profile to get the top k.

4.5 Goal Modeling

The three basic "operations" that are performed by all our recommendation mechanisms is to form: (a) the *goal space* $\mathcal{GS}$(A), (b) the *goal implementation space* $\mathcal{IS}$(A), and (a) the *action space* $\mathcal{AS}$(A), given an activity, i.e., a set of actions A. Each mechanism uses them accordingly (see the algorithms in Subsections 4.4.1, 4.4.2 and 4.4.3). Performing these

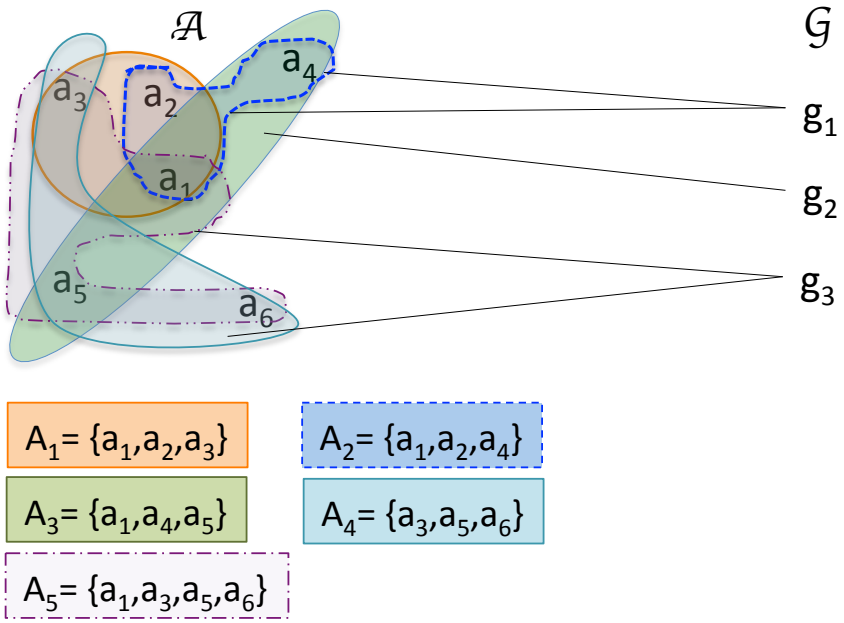


Figure 4.2: Illustration of the Association-based Goal Model.

Index	**Description**
A-idx	*Action Index* matches each existing action to an id
G-idx	*Goal Index* matches each existing goal to an id
GI-AV-idx	*Goal Implementation ActiVity index* gets the action set (i.e., activity) of each goal implementation id, this id is the goal implementation id (pId) where it belongs
GI-G-idx	*Goal Implementation Goal Index* gets the goal id (gId) of a goal implementation
A-GI-idx	*Action to Goal Implementation Index* gets for each action a all the implementations where a participates

Table 4.1: Indexes for goal-based recommendation

operations considering a goal implementation set with a couple of goal implementations such as the one in the Example 9, is not a demanding task. However, when moving to hundreds or millions of implementations, the cost gets prohibitive. For instance, to form the user goal space, one should visit one by one millions of implementations and check whether there exist any actions from the user activity in their activity.

Therefore, we should find a way to *efficiently retrieve the goal, implementation and action spaces considering the associations among actions and goals through the goal implementations*

We suggest a model that sees each activity A in the goal implementation set L as a *hyper-edge* that connects the actions that participate in it. Moreover, it labels each

activity A with the goal that fulfills given a goal implementation $\langle g, A\rangle$. Figure 4.2 graphically illustrates this model that will be referred to as *association-based*, considering the goal implementation set in the Example 9. The association-based goal model captures the associations *between actions and goals*.

Model implementation. We implement the *association-based* goal model using a number of indexes. These indexes allow us to retrieve the information we need in real time.

We first build an index *A-idx* for the action set and an index *G-idx* for the goal set. The *aId* and *gId* refer to the ids that correspond to action a and goal g, respectively. Keeping the information derived from the goal implementation set L needs a more complex structure. We refer to each goal implementation using a unique identifier id. We split the information of the goal implementation pairs in two indexes: *Goal Implementation ActiVity index* (*GI-AV-idx*) and *GI-G-idx* (*Goal Implementation Goal Index* (*GI-G-idx*). The first one matches the activity of a goal implementation to the id of the goal implementation where it belongs. We store a set with the ids of the actions. The second index matches each goal id to all the implementation ids that exist for the specific goal. Now we need to connect the goal implementations with the actions they contain. For this, we use *A-GI-idx* (*Action to Goal Implementation Index*) that retrieves all the goal implementation ids where an action contributes, i.e., the implementation ids (pIds) s.t., $a \rightsquigarrow^p g$.

Equations 4.10, 4.11, and 4.12 describe how we exploit the above index structures to implement the three basic operations that we described earlier, i.e., to form the goal, implementation and action space given an activity.

$$\mathcal{GS}(A)=\{GI\text{-}G\text{-}idx[pId] \mid a\in A \wedge aId=A\text{-}idx[a] \wedge pid=A\text{-}GI\text{-}idx[aId]\} \tag{4.10}$$

$$\mathcal{IS}(A)=\{A\text{-}GI\text{-}idx[aId] \mid a\in A \wedge aId=A\text{-}idx[a]\} \tag{4.11}$$

$$\mathcal{AS}(A)=A-\{GI\text{-}AV\text{-}idx[pId] \mid a\in A \wedge aId=A\text{-}idx[a] \wedge pid=A\text{-}GI\text{-}idx[aId]\} \tag{4.12}$$

4.6 Query Answering on the Association-based Goal Model

The indexes that are used in the implementation of our association-based goal model (ref. Table 4.1) have been selected based on the requirements of the goal-based recommendation mechanisms. However, this model is a very rich source of knowledge about the goal

implementation set. It can be exploited to retrieve information for different types of queries. Interesting types of queries include the following.

First of all, given an action a, one type of query is to find: *(Q1)* "Which goals can benefit from action a?". These queries can be answered very simply by returning the goals in the goal space $\mathcal{GS}(a)$ of the action. For instance, in Figure 4.2 the goals that can benefit from a_4 are the g_1 and g_2, while all the goals can benefit from a_1 (ref. Figure 4.2). Respectively to the goal space of an action a, its action space $\mathcal{AS}(a)$ contains a number of actions that are associated with a. Going one step further, we can define an *action relatedness* and rank the actions in the action space. One option is to use the goal-based action representation that was introduced in the *Best Match* strategy using for the vector the goals in the goal space $\mathcal{GS}(\mathcal{AS}(a))$. This way, we can answer queries of the type: *(Q2)* "Which actions are related to action a?". For instance, the fact that the actions a_2, and a_4 appear in the implementations of goals g_1 and g_2 and not in any implementation of g_3 shows that a_4 is more related to a_2 than the rest of the actions in $\mathcal{AS}(a_2)$ (ref. Figure 4.2).

All the previous queries are referring to actions. However, such queries can be answered for goals as well, i.e., queries of type: *(Q3)* "What are the different sets of actions that should be performed to fulfill goal g"? and *(Q4)* "Which goals are related to goal g?". For the first type of queries an additional index in needed: the G-GI-idx index that gets the goal implementations for every goal, i.e., *the goal implementation space of the goal* (respectively to the goal implementation space of an *action*). After the retrieval of the implementations, the GI-AV-idx index that matches the goal implementations to their activities can be used to retrieve the alternative action sets. The second type of queries (*Q4*) can be answered by evaluating the goals that share common actions with the current goal in their implementations. The common subsets of actions in the implementations of the goals would be one way of answering the query. Alternatively, a distance-based measure on an action-based representation of goals can be employed.

The different types of queries and the alternative ways to answering them indicates that *the knowledge that can be leveraged through the association-based model* can be exploited in a lot of services and applications other than recommendations.

4.7 Experimental evaluation

For our experimental evaluation, we examine two different scenarios: (*a*) a grocery store where clients (users) buy food products, and (*b*) a system where users record actions they perform in their lives such as *read a book* or *eat healthy*. We selected these scenarios to show that goal-based recommendation can be used both in practical scenarios where existing recommendation techniques have already been applied, and to offer innovative services that have not been available so far. In both scenarios, we want to recommend to the users *actions of interest* (i.e., buy + "a product" and everyday actions respectively). The actions are characterized to be of interest based on the goals that they serve: food products can serve *food recipes*, while everyday actions can serve *life goals* such as *lose weight* or *learn english*. Another reason for examining these scenarios is that they cover two different cases: the first one covers the case where the same action participates in a great range of goal implementations (on average 1.2K impl.), while in the second case, most actions are limited to specific "families" of goals (on average an action participates in 3.85 implementations).

Dataset Description. The first dataset is an open source grocery shopping dataset (https://github.com/julianhyde/foodmart-data-mysql) that contains *1560 food products* (items) and records of customer purchases in different time slots, i.e., *carts*. The food products are organized in 128 (sub)classes such as "baking goods", "seafood", "fruit", "spices" and so forth. Clients can utilize these products in various *recipes* to produce *different dishes* (goals). We used a dataset of *56498 recipes* from a food ontology (http://data.lirmm.fr/ontologies/food#Recipe). Based on the description of each food product, we matched each product to an ingredient leaving out products that are not included in any recipe, such as napkins. Therefore, each cart can be seen as the *user activity*, the set of recipes as the *goal implementation set* L, while the *actions* refer to the purchase of certain products/ingredients. We examined *20522 user activities*. The number of implementations in which an action participates on average, that will be referred to as *connectivity*, is 1.2K.

The second dataset consists of goal implementations from a *goal-setting online social platform* called 43Things where users could publish the goals they set in their lives, "cheer" other users' goals and efforts, and provide descriptions about how they managed to fulfill their goals. We have extracted *18047 goal implementations* that contain *3747 goals* such *pay my depts*, *get a new job*, *lose weight*, and *5456 actions* e.g., *stop eating at restaurants* and *drink more water*. Both goals and actions are identified by unique identifiers. In contrast to the foodmarket dataset, users are focused on a few real life goals. In total, the users are 8071. The majority of the users (5047 users) are pursuing one goal, 1806 of them pursue 2 goals, 623 pursue 3, and 595 pursue more than 3 goals. Moreover, the

action *connectivity* is very low, 3.84. The fact that actions here in contrast to actions that involve food ingredients are useful in a narrow range of goals and by extension goal implementations makes the analysis of the two sets more intriguing. We examined *8071 user activities* by taking all the actions that each user has performed for fulfilling all her/his goals, shuffled the derived set, and hide the 70%. To examine the completeness of the corresponding goals after the user has performed the recommended actions, we respectively created the user activities by hiding the 30% of the actions. These activities are both about goals that are close to fulfillment and about goals that are still hidden (there is no clear evidence about them).

Methods. Beyond the goal-based mechanisms, we examine how the two main item recommendation approaches, namely *Collaborative* and *Content*-based filtering behave under the same context. In order to make clear the differences between the state-of-the-art approaches and the approach we suggest, we consider a pure content-based and a pure collaborative filtering method. For the Collaborative Filtering method, we used a memory-based approach that computes the most similar user activities to the current user's activity [Sarwar et al., 2001]. The fact that the users do not rate the items but we have *selection*, *non-selection* allows us to use jaccard distance for forming the user neighborhoods avoiding the vector sparsity problem. For the Content method, in order to build the user profile and evaluate the candidate actions performing the comparison between the profile and the vectors representing the actions, domain-specific features are needed. For the foodmarket dataset the domain-specific features are the 128 (sub)categories of the food products (e.g., "baking goods", "seafood"). On the other hand, for the 43T dataset, there are no widely accepted domain-specific features; therefore, we do not apply the content approach. For the goal-based recommendations, we used the methods described in Subsections 4.4.1, 4.4.2, and 4.4.3), namely $Focus_{cmp}$ and $Focus_{cl}$, *Breadth* and *Best Match*.

Subsection 4.7.1 compares all methods. Subsection 4.7.2 focuses on the time efficiency of the goal-based methods.

4.7.1 Evaluation and Comparisons

Since we are introducing a novel recommendation approach, in Subsection 4.7.1, we first verify that this approach, indeed offers a different perspective to the users. To do so, we perform several *comparisons* on the results (i.e., the recommendation lists) produced by all the goal-based and the standard recommendation mechanisms.

- We compare the lists formed by the goal based mechanisms with the two standard recommendation mechanisms (ref. C.1.1. *Result Overlapping*).

- Collaborative filtering is based on the past activities of similar users (to the current user), while our algorithm is intended for discovering useful actions, i.e., actions that will help the user fulfill one or more goals. Thus, we examine whether actions that appear frequently *in the activities of other users* (popular actions) appear frequently *in the recommendation lists* as well. In other words, we study which recommendation mechanisms perpetuate the collective user behavior (ref. C.1.2. *Correlation of appearances in the user activities and the respective recommendation lists*).
- Next, we examine how useful the actions in the top-10 lists of each algorithm are for the user. To measure *usefulness*, we estimate the completeness of the goals in the user's goal space after s/he performs the recommended actions (ref. C.1.3. *Usefulness*).
- We also study how *similar* the recommended actions in each list are presenting their (max, min and avg) pairwise similarity based on their domain-specific characteristics. Retrieving items that are very similar to each other is often considered a drawback of the Content-based filtering. It is important to understand how the rest of the examined approaches work as well (ref. C.1.4. *Pairwise similarity of the recommended actions*).
- Moreover, we examine how many of the actions in the recommendation lists have been indeed performed by the users. These actions are not of course part of the considered user activity but the users "like" them since they have performed them at some point (ref. C.1.5. *Average Percentage Of Recommended Actions that the user has indeed Performed.*)

Subsequently, Subsection 4.7.1 further examines the actions retrieved by the goal-based mechanisms (ref. C.2.1 *Frequency of Retrieved Items*) and presents the percentage of common actions in their top-10 recommendation lists (ref. C.2.2 *Result Overlapping of Goal-based methods*).

Comparison of all Approaches

C.1.1. Result Overlapping. Table 4.2 illustrates a very low overlapping of the top-10 lists formed by the goal-based mechanisms with the lists formed by the two state-of-the-art approaches. This result is expected, since as we have explained in Section 4.2, these approaches adopt fundamentally different philosophies.

C.1.2. Correlation of the number of appearances in the user activities and the number of appearances in the respective recommendation lists of the *top-20 most popular actions.* Table 4.3 illustrates the Pearson's correlation between these two numbers. Correlation takes negative values from -1 to 1, with 1 reflecting highly correlated values. Collaborative

	Food Market		*43T*
Methods	*Overlap.with Content Filt.*	*Overlap. with Collab. Filt.*	*Overlap. with Collab. Filt.*
Best Match	0.014%	0.0114%	0.17%
$Focus_{cmp}$	0.012	0.0064%%	0.11%
$Focus_{cl}$	0.005%	0.0084%	0.13%
Breadth	0.0142%	0.0114%	0.3%

Table 4.2: Overlap of the top-10 recommendation lists produced by the goal-based mechanisms with the lists produced by the standard recommendation approaches.

	Food Market	*43T*
Methods	*Correlation*	*Correlation*
Content	0.115	-
Collaborative	0.35	0.75
Best Match	-0.13	-0.24
$Focus_{cmp}$	-0.048	-0.26
$Focus_{cl}$	-0.02	-0.27
Breadth	-0.04	-0.15

Table 4.3: How correlated are the recommendation lists with the top-20 popular actions in the user activities.

filtering, which looks into the past actions of similar users for actions that may be of interest to the current user, shows the highest correlation. On the other hand, goal-based methods show negative correlation. They do not promote actions that were popular (frequent) so far. The content-based approach shows a lower correlation than collaborative filtering, which is still high in comparison to the goal-based methods.

C.1.3. Usefulness: the completeness of the goals in the user's goal space after s/he follows the recommended actions. The actions recommended to the user can help her get closer to the fulfillment of (or fulfill) one or more goals. Table 4.4 shows the average average (AvgAvg), min (MinAvg) and max (MaxAvg) completeness values for all the recommendation lists formed for the two datasets. These values are estimated by finding first the average, minimum and maximum values of completeness of all the goals that are related to the user *considering each list separately.* Subsequently, the average for all the recommendation lists is estimated. Moreover, Figure 4.3 shows graphically the AvgAvg values. The goals that we consider in the estimation of goal completeness in the case of the 43T are those that the user has added in the system, while in the case of the food market we consider the whole user's goal space since we do not have any information about which goals the user is pursuing in reality. In the foodmarket dataset, the goal implementation

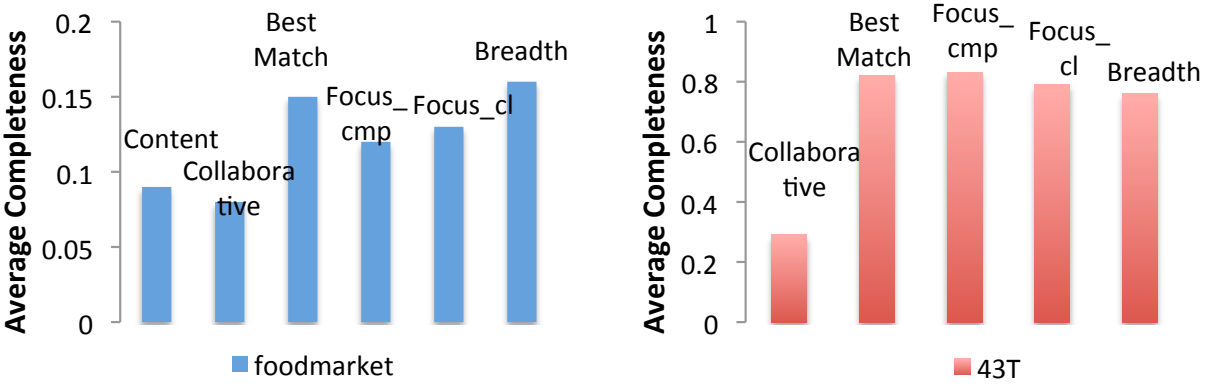


Figure 4.3: The *average goal completeness* per list *after the user follows the recommended actions.*

Methods	*Food Market Completeness*			*43T Completeness*		
	Avg-Avg	*Avg-Max*	*Avg-Min*	*Avg-Avg*	*Avg-Max*	*Avg-Min*
Content	0.09	0.67	0.05	-	-	-
Collabo-rative	0.08	0.63	0.05	0.29	0.32	0.26
Best Match	0.15	0.79	0.076	0.82	0.87	0.77
$Focus_{cmp}$	0.12	0.73	0.064	0.83	0.88	0.77
$Focus_{cl}$	0.13	0.74	0.062	0.789	0.84	0.73
Breadth	0.16	0.8	0.076	0.76	0.8	0.72

Table 4.4: How complete become the goals of the user *after s/he follows* the actions in the recommednation list that was formed based on her/his activity.

space can be large and not every goal can be fulfilled by performing only 10 actions (i.e., the actions in the recommendation list) in any case. As a consequence, the AvgAvg values in this dataset are not that informative in comparison to those of the 43T dataset.

We observe that *Breadth* and *Best Match* in the first dataset and $Focus_{cmp}$ in the second dataset manage the largest completeness (considering both the user activity and the recommended actions), while the lowest contribution is met in the state-of-the-art algorithms. The results are explained by the fact that *Best Match* considers the whole user's goal implementation space, *Breadth* creates a well-connected subspace, while $Focus_{cmp}$ selects a single goal (actually a single implementation), if possible, and extends to a few more to complete the recommendation list. *If the user wants to get closer to a wider range of goals, s/he should select Breadth; otherwise (i.e., if s/he is focused on a few goals), s/he should select* $Focus_{cmp}$. *Best Match* and $Focus_{cl}$ follow.

Methods	*Pairwise Action Similarity*		
	AvgAvg	*AvgMax*	*AvgMin*
Content	0.81	1	0.6
Collaborative	0.16	0.5	0.05
Best Match	0.33	0.72	0.22
$Focus_{cmp}$	0.24	0.31	0.21
$Focus_{cl}$	0.24	0.34	0.19
Breadth	0.33	0.73	0.22

Table 4.5: Pairwise similarity (based on the features of the products) among the actions within each recommendation list for the foodmarket dataset.

C.1.4. Pairwise similarity of the recommended actions (i.e., the corresponding products) in each list. Table 4.5 shows the pairwise similarity among the retrieved actions in each recommendation list. Due to the lack of widely-accepted domain-specific characteristics for the actions in the 43T dataset, we study the food market dataset. AvgAvg is estimated in two steps: first *the average pairwise similarity* considering all the action pairs within each list is estimated, and then the average of the derived values is estimated. The same applies for AvgMax and AvgMin. As expected *Content* shows the highest value with an AvgAvg pairwise value 0.8 and AvgMin value 0.6. Collaborative filtering shows the lowest similarity (AvgAvg 0.15), while all goal-based mechanisms are found in the middle (avg-avg: 0.24-0.33). However, looking at the average maximum pairwise values (AvgMax), we see that the goal-based methods *Best Match* and *Breadth* often share a pair of very similar actions in their lists (on average their max pairwise similarity values are 0.72 and 0.73 respectively). The two *Focus* methods are the goal-based methods that retrieve highly dissimilar actions in most of the cases.

C.1.5. Average percentage of recommended actions that the user has indeed performed (per recommendation list). In the food market dataset, we consider as the user's current activity a single cart; we have more than one cart for the same user in different time slots though. On the other hand, in the 43T dataset we consider only the 30% of the actions that the users have performed to fulfill their goals. Therefore, we can check whether the different techniques by considering only the actions in the user activity, recommend actions that the user has performed. We should clarify that the average percentage of recommended actions that the user has indeed performed does not reflect the precision of the recommendation tasks since the user has not acted after checking the recommendation lists. In fact, it shows the percentage of the recommended actions for which the user has shown interest at some point. Unlike precision, being able to retrieve actions that the user would anyway perform can be an advantage or a disadvantage for a

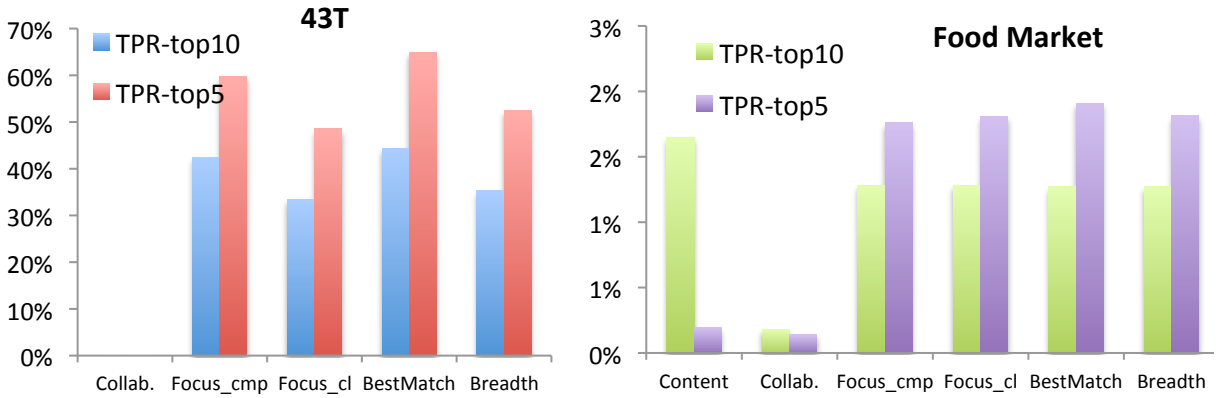


Figure 4.4: Percentage of recommended actions that the user has indeed performed (True Positive Rate for top-5 and top-10 lists).

recommendation technique depending on the view point of the application. If the purpose of the recommendation system is to show to the user unknown actions as well, a very high percentage is not preferable. On the contrary, if the purpose of the system was to provide the user with a discount coupon in order to keep her/him satisfied, a high value would be preferable. Keeping that in mind, we can say that the average percentage represents the Average True Positive Rate. Figure 4.4 illustrates for each method the *Avg TPR* for top-5 and top-10 lists. In the top-5 lists, first the *Best Match*, then the $Focus_{cmp}$ and *Breadth* show the largest percentage. In the top-10 lists of the foodmarket dataset though, it is the *Content* method that shows the highest percentage. Nevertheless, all the methods show low percentages in the foodmarket dataset. This is explained by the fact that we have no more than 3 carts for each user.

Further Comparison of Goal-based results

Considering the lists derived from the goal-based methods, we have already argued about the fact that the appearance of an action in the recommendation lists is not correlated to its appearance in the user activities (ref. Table 4.3). Next we also present whether there exist actions that monopolize the recommendation lists, and how different are the recommendation lists formed by the alternative goal-based methods (*Result Overlapping of Goal-based methods*).

C.2.1. Frequency of Retrieved Items. In recommenders, we do not want certain actions to monopolize the recommendation lists. In the 43T dataset, the frequency of an action in different recommendation lists is very low: *at maximum 0.001*. On the other hand, in the food market dataset, where there are a lot of actions that participate in a great number of implementations (average connectivity 1.2k), the frequency is higher. Figure 4.5 illustrates that the majority of actions appear with frequency less than 0.2. However, *Best Match*

and then *Breadth*, in their effort to serve more than one goal at the same time, repeat the same actions in more recommendation lists (22% and 14% actions respectively with frequency above 0.2). *The actions with high frequency are those that appear frequently in subsets of implementations that share common actions.* Actions that appear in many goal implementations *but together with different actions in each goal implementation* are not selected more frequently. On the contrary, Figure 4.6 shows that very few actions that appear frequently in the goal implementation sets are in the end selected by any goal-based mechanism. The great majority (more than 92%) of the retrieved actions (by all the goal-based mechanisms) appear in the implementation set with a frequency less than 0.2.

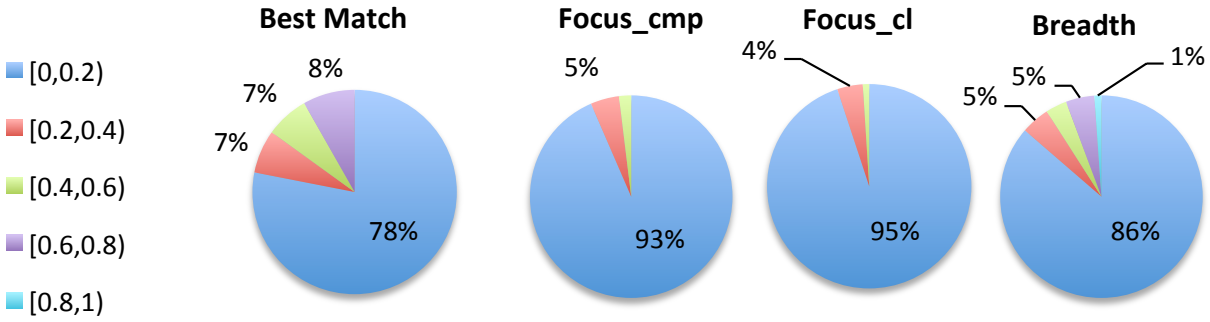


Figure 4.5: How often the same action appears in the recommendation lists that have been formed for the user activities of the food market dataset. Distribution of actions in frequency ranges.

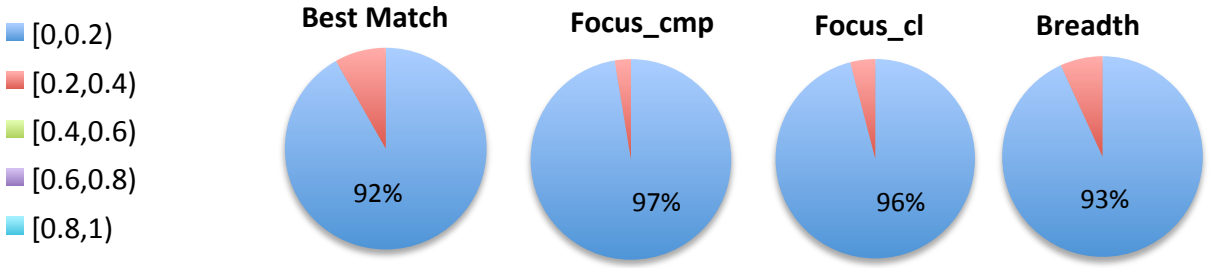


Figure 4.6: How often the same retrieved action appears in the goal implementation set (herein recipes). Distribution of actions in frequency ranges.

C2.2. Result Overlapping of Goal-based methods. In Paragraph *C1.1*, we have presented how different are the results of the goal-based mechanisms from those of the standard recommendation methods, next we present the result overlapping of the goal-based mechanisms. Table 4.6 illustrates the percentage of common actions in their top-10 lists considering again as input the 21k real carts and the 8k user activities of the food market and the 43T datasets respectively. First of all, we observe a great overlapping in the results of *Best Match* and *Breadth*: 98% and 79% respectively. The overlapping is higher

	Food Market	*43T*
Methods	*Overlapping*	*Overlapping*
Best Match-$Focus_{cmp}$	42%	68%
Best Match-Breadth	98%	79%
$Focus_{cmp}$-Breadth	44%	71%
$Focus_{cl}$-$Focus_{cmp}$	35.6%	78%
$Focus_{cl}$-Best Match	49%	72%
$Focus_{cl}$-Breadth	49%	72%

Table 4.6: Common actions in the top-10 recommendation lists of the goal-based mechanisms.

in the first case because in the food market ingredients participate in a lot of recipes at the same time. Therefore, *Breadth* instead of examining subsets of the user's goal space to evaluate a certain action, it ends up considering (almost) the whole goal space similarly to *Best match*. In general, the user profile that *Best Match* considers reflects more strongly her/his preference towards a subset of goal(s); thus it (almost) neglects the rest of the goals in the user's goal space the same way *Breadth* does. Since the two algorithms show similar behavior, *Breadth* is preferred since, as we will see in Subsection 4.7.2, *Breadth* is significantly more efficient in terms of time.

Moreover, $Focus_{cmp}$ and $Focus_{cl}$ retrieve the same actions in 35.6% and 78% of the lists respectively. In these cases, there exist goal implementations for which the user has performed most of the actions (completeness) and at the same time these are the implementations with the less remaining actions. Furthermore, $Focus_{cl}$ and $Focus_{cmp}$ show an overlapping of over 40% and 70% (for the respective datasets) with *Breadth* and *Best Match*. This is justified by the fact that the Focus mechanisms after popping out all the actions of the goal implementation on which they have selected to focus, they move on to another goal implementation. Therefore, they select actions from different goal implementations as *Breadth* and *Best Match* do. Another way to see this is that the latter two algorithms select actions that serve more than one goals at the same time; but that means that the selected actions serve each single goal on its own as well.

Another observation is that the overlapping in the lists for the 43T dataset is larger than in the lists for the food market dataset in all the cases because the action space of the users are wider in the latter dataset due to the high action connectivity. Considering a larger set of candidate actions, the algorithms are not forced to select the same actions due to lack of alternatives.

4.7.2 Scalability

We ran the 4 goal-based strategies (i.e., the 3 strategies plus the extra option for *Focus*) considering as input each of the *real user activities*, i.e., the 20522 carts of the food market dataset described in the beginning of Section 4.7, and 8 implementation sets of different characteristics: (a) implementation set size, (b) action set size, and (c) number of implementations in which an action participates on average (*connectivity*). Table 4.7 describes the characteristics of the sets used in the evaluation. The first implementation set (IS) consists of the goal implementations that correspond to the recipes from the food ontology and the other three are variations of the original recipes. The sets IS_2 (280K impl), IS_3 (565K impl) and IS_4 (1.6M impl) have been generated by keeping the second parameter (action set size) stable and increasing the first one (implementation set size). The third parameter (i.e., connectivity) increases respectively. The sets IS_5 and IS_6 have been generated by keeping the implementation set size stable and increasing the action set size but keeping the connectivity in a relatively low value (449-547); and the sets IS_7 and IS_8 considering the action sets that have been used for IS_5 and IS_6 but with a higher participation in the implementations, i.e., a very high connectivity value (11.6K-12K).

Results. Table 4.8 reports the average time per information need (i.e., per user activity) in secs and Figure 4.7 illustrates it graphically (in millisecs). We observe that the *Best Match* shows the highest execution times in all the cases. The reason is that in goal-based profiles the feature space is not fixed, and thus the representation of the actions is formed on the fly. The rest of the mechanisms show low recommendation time even in the extreme cases of the sets IS_4 and IS_8 (connectivity: 50315, average participation: 19M, and connectivity: 12110, average participation: 137M respectively). The difference between $Focus_{cl}$ and $Focus_{cmp}$ results from the two set operations that the mechanisms

Set	*Avg Connectivity*	*Num Of Distinct Actions*	*Avg Participation*	*Num Of Implementations*
IS	1.2K	380	464.4K	56K
IS_2	7.6K	380	2.9M	282K
IS_3	15.6K	380	6M	564K
IS_4	50.3K	380	19.2M	1.6M
IS_5	547	3.7K	2M	56K
IS_6	449	11.4K	5.2M	56K
IS_7	11.6K	3.7K	42.8M	56K
IS_8	12.1K	11.4K	138M	56K

Table 4.7: Goal Implementation Sets.

Alg	IS	IS_2	IS_3	IS_4	IS_5	IS_6	IS_7	IS_8
Best Match	.37	1.5	3	9.7	.57	11.1	10	34.8
$Focus_{cl}$	.001	.053	.096	.35	.008	.69	.5	1.6
$Focus_{cmp}$	.091	0.42	0.86	2.98	.061	.36	.8	2.7
Breadth	.006	.089	0.089	.34	.009	.16	.95	4.1

Table 4.8: Average Execution Time in secs.

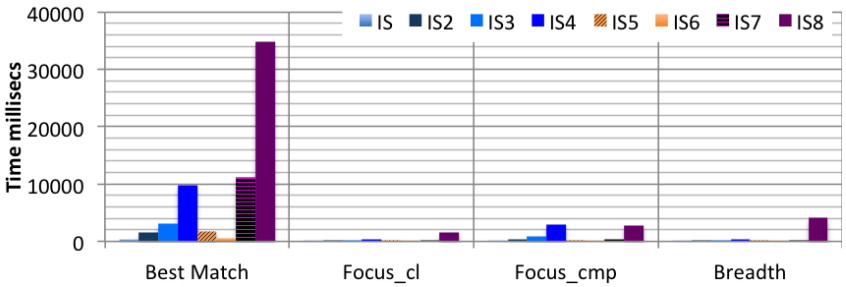


Figure 4.7: Average recommendation time considering implementation and action sets with different characteristics (ref. Table 4.7).

use, i.e., asymmetric difference and intersection respectively.

The execution time of all methods increases when the average participation (i.e., the average number of actions in the goal implementations) increases. This happens in two cases: (a) when the number of implementations increases and the existing actions participate in more implementations, and (b) when the number of actions increases but at the same time the average participation increases, i.e., the average activity size of the implementations, increases. *The number of actions alone does not affect much the execution time, it is the higher average participation that results in higher execution times.* For instance, we see that although IS_5 contains more actions than IS_2, the execution times when the IS_5 is considered are significantly lower comparing to the IS_5.

Chapter 5

Building Goal Implementation Sets from text descriptions

In Chapter 4, we have made a short reference to the extraction of goal implementation sets from text sources (ref. Section 4.3). In this chapter, we are dealing with the problem of *goal implementation extracton* from textual descriptions that includes the identification and modeling of the described actions and the goals that they fulfill (implement).

- In particular, we introduce and formally define the problem of goal implementation extraction.
- We provide a method for identifying the expressions in the text that may refer to some action and for deciding which of these expressions actually correspond to actions, based on syntactic analysis and learning.
- We provide algorithms for grouping different expressions referring to the same action together to form a unique action. This decision is based on similarity comparisons in the spirit of entity identification comparisons.
- We provide an adapted version of the required comparisons in clustering that avoids the complexity problems that the N×N item comparisons require, and a different version of the centroid that better serves our goals.
- We illustrate the effectiveness and scalability of our approach through evaluation with a real dataset.

In what follows, we introduce the problem in Section 5.1, motivate the problem in Section 5.2 and formally introduce it in Section. 5.3. In Section 5.4, we present our action recognition and identification algorithms and goal modeling approach respectively. We present our evaluation results in Section 5.5. The related work that regards the extraction of information about goals or other related concepts such as actions and intentions has been presented in Section 2.2.5.

5.1 Introduction

Information extraction from text and the organization of the extracted knowledge in structured ontologies, such as YAGO [Suchanek et al., 2008], Kylin/KOG [Wu and Weld, 2007], and DBpedia [Auer et al., 2007] that model the real world as expressed in the text are well-studied problems. However, in our problem there is no need of tasks such as entity and relationship identification [Ritter et al., 2011], or event detection methodologies [Weng and Lee, 2011], we focus on phrases that describe actions without considering aspects such as the entities that perform the actions, which conditions should be met to perform an action, or any ordering information. The main objective of our extraction task is to recognize those phrases (i.e., phrases about actions) in different textual descriptions so as to detect the associations among different goals. Having detected the associations we can build a goal model such as the association-based model in Figure 4.2 (ref. Section 4.5). In practice, actions consist of groups of phrases. However, in the end, actions and goals are represented by a unique identifier; no other information is kept.

In our goal implementation extraction mechanism, the first step is the *action chunk recognition*, i.e., the recognition of word sequences (chunks) that describe actions. Subsequently, since the same action can be used towards the fulfillment of different goals, we need to identify which chunks refer to the same action. We call this step *action identification* to emphasize its similarity to the traditional task of entity identification (a.k.a., duplicate detection, synonym identification, etc) in information extraction, where textual expressions referring to the same real-world object are identified and assigned a unique id that models the respective object. The final step is *goal implementation modeling*, i.e., modeling the goals alongside their implementation as sets of actions.

We are interested in goal implementation extraction from user-generated text, such as posts from 43Things, mylifelist[1], where users describe their actions towards a goal in free-form text. Such text poses several challenges compared to more structured documents such as the "how-to" pages of WikiHow, whose rigid form (e.g., actions are given in a numbered list) allows straightforward extraction of the actions. *First*, there are no guarantees on the expressions that have been used in the text for describing an action or on the text format. The text may not use standard phrases, such as "you should" or "it is needed", or special symbols, such as html tags, bullets or numbering. Furthermore, a text segment describing an action should be self-explanatory and at the same time it should not contain redundant content such as terms that refer to other actions or non-actions. As the size and the format of the chunks that describe actions vary, no strict rules can be imposed for action chunk recognition. Neither can we segment a text using standard text

[1] http://www.mylifelist.org

units such as n-grams or sentences. *Second*, action chunks referring to the same action may have different forms and expressions making action identification hard. In typical information extraction, entity identification can be performed by comparing the extracted entities (e.g. "president Obama", "Barack Obama") to a well-known ground truth, such as wikipedia entities. For actions, there is no such ground truth. Thus, our methods can only be based on the dataset used with no prior knowledge. *Third*, a goal can be fulfilled through different sets of actions and the same action can be used towards the fulfillment of different goals making goal modeling challenging. In this work we aim at providing solutions towards these challenges.

5.2 Motivation

The 43things site is a site where users share their experiences on how they have achieved a number of goals in their real life. Consider the three posts shown at the top of Figure 5.1. In the first post, user KellyBelle describes how she achieved to buy a car through a set of actions that allowed her to find the required amount of money and make the right choice. In the second post, user Ninn077 also describes how he achieved to buy a car, while in the third post, author Mom189 talks about how she achieved to buy her own house.

The posts contain a number of expressions (underlined for easy identification) that describe actions that the author has performed. Different expressions may correspond to the same action. For instance, the expression *"got a loan from the bank"* in the second post and the expression *"got a loan"* in the third refer to the same action (id: a_8). The table in Figure 5.1 shows these expressions alongside an id that corresponds to the action that each expression describes. We also observe that the same goal can be achieved through different sets of actions, and the same action may be performed towards the fulfillment of different goals. For instance, the action with id a_8 (*get a loan* or *get a loan from the bank*) has been used for getting a car (goal g_a) and for buying a house (goal g_b), while the action a_1 about searching on the Internet is met in both posts about goal g_a.

Having extracted the knowledge about actions organized alongside the goals they fulfill enables a better understanding of goal implementations and the interrelationships between actions. One can now ask queries such as "how can I get a car" (goal g_a) and get as an answer the alternative sets of actions: $\{a_1,a_2,a_3,a_4,a_5,a_6\}$, $\{a_1,a_7,a_8,a_9\}$. Moreover, we could retrieve the different goals that one can fulfill by *search on the Internet*(a_1), or answer whether and how *move from parents' house*(g_b) and *get a car* (g_a) are related. For the latter we may get as an answer the common actions that one can perform to fulfill both goals (i.e., *get a loan*).

get a car *(KellyBelle)* [14 Aug 2011]

I have been searching on the Internet for a good opportunity. I asked a friend and he told me to search at craigslist. I had my dad helping me to choose the right car and also negotiate the price.

get a car *(Ninn077)* [18 Nov 2012]

Finally I got a car!!!. I searched on the internet and found the car of my dreams. I got a loan through the bank and used my payroll account for paying it. The bank was a lot more willing to give me the loan that way.

move from my parents home *(Mom189)* [5 Nov 2011]

My husband and I decided to buy our own house. I found a second job. It was just for the weekends but it helped us to save up some money. We also got a loan from the bank. We have been biding on every house that we liked in the area but in the end we moved in to our new home!!

Goal	ActId	Expression
g_a	a_1:	searching on the Internet
	a_2:	asked a friend
	a_3:	searched at craigslist
	a_4:	dad helped
	a_5:	choose the right car
	a_6:	negotiate the price
g_a	a_1:	searched on the internet
	a_7:	found the car
	a_8:	got a loan
	a_9:	used payroll account
g_b	a_{10}:	buy our own house
	a_{11}:	found a second job
	a_{12}:	save up some money
	a_8:	got a loan from the bank
	a_{13}:	biding on every house
	a_{14}:	moved in to our home

Figure 5.1: Three example posts from 43things and the respective actions and goals.

5.3 Problem Statement

We consider a set $\mathcal{T}$ of textual descriptions provided by users, a set $\mathcal{G}$ of goals and a set $\mathcal{A}$ of actions. A textual description $t \in \mathcal{T}$ describes how a user has achieved or proposes to achieve a goal $g \in \mathcal{G}$. When users provide a description, they also specify the goal that this description is about. Such pairs are called *user proposals*.

Definition 13 *A* user proposal *is a pair* $\langle g, t \rangle$ *of a goal* $g \in \mathcal{G}$ *and a description* $t \in \mathcal{T}$.

We denote the set of user proposals as $\mathcal{P}$.

Let $\mathcal{S}$ be the set of all sequences of words defined from the descriptions in $\mathcal{T}$ and c be a word sequence. For two word sequences c, c', the operator $c \triangleright c'$, indicates that c is a *subsequence* of c'. In a description t, certain word sequences refer to actions. We assume a function $ref|\mathcal{S} \rightarrow \mathcal{A} \cup \{null\}$, that given a word sequence $c \in \mathcal{S}$, returns an action $a \in \mathcal{A}$ if c refers to the action a, while any subsequence of it does not, otherwise it returns *null*. A sequence c that refers to an action is called *action chunk*. We denote the set of all action chunks as $\mathcal{C} = \{c \mid c \in \mathcal{S} \wedge ref(c) \in \mathcal{A}\}$. Users may refer to a specific action in different ways, thus, there may be more than one $c \in \mathcal{C}$ for which $ref(c) = a$, for the same $a \in \mathcal{A}$.

Definition 14 *A* goal implementation *is a pair* $\langle g, A \rangle$, *where* $g \in \mathcal{G}$ *and* $A \subset \mathcal{A}$.

Problem Statement. In a description of a user proposal, there are typically more than one subsequences that are action chunks. Our goal in this work is to identify these action chunks, the actions to which they refer, and combine these actions in a set to form goal implementations. More specifically, given a set $\mathcal{P}$ of user proposals, our goal is to identify the set $\mathcal{I} = \{\langle g, A \rangle | \ \langle g, t \rangle \in \mathcal{P} \wedge \exists s \triangleright t \colon ref(s) = a \ \forall \ a \in A\}$.

5.4 Goal Implementation Extraction

Figure 5.2 graphically depicts the main steps of the goal implementation extraction. *Action chunk recognition* extracts the text parts that describe actions, i.e., the action chunks (Subsection 5.4.1). *Action identification* groups textual expressions referring to the same action and assigns to them a unique identifier that models the respective action (Section 5.4.2). Once the final assignment has been concluded, each proposal is converted into a goal implementation.

5.4.1 Action Chunk Recognition

In goal implementation extraction, we are not dealing with generic web content but with content where users describe goals and actions towards these goals. Hence, a textual

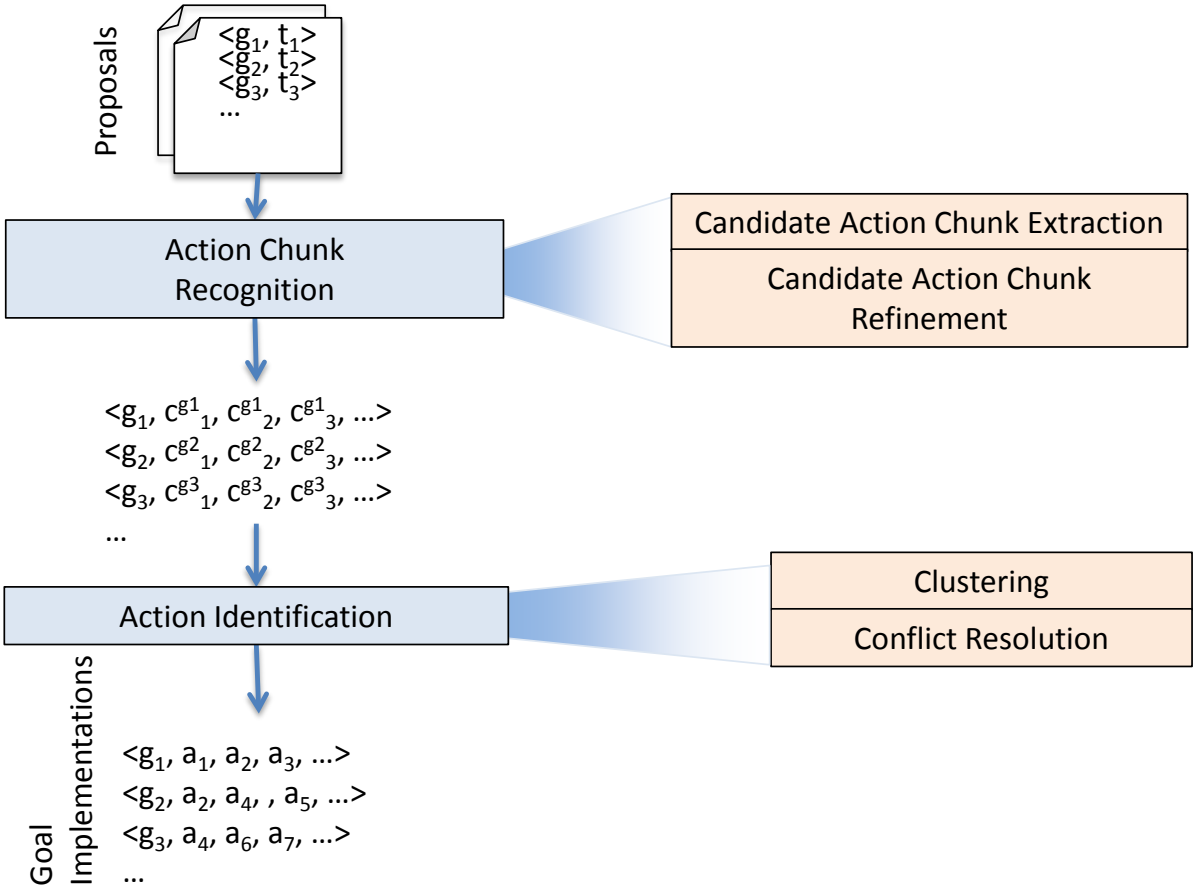


Figure 5.2: The goal implementation extraction process.

description t of a proposal $\langle g, t\rangle$ (e.g., the posts in Figure 5.1) does not contain passages with completely irrelevant content that could be discarded beforehand.

To identify the action chunks in the textual description t of a proposal $\langle g, t\rangle$, rules consisting of terms and syntactical information have been used in the past [Castellanos et al., 2012; Strohmaier, 2008]. These rules were hand-crafted and were either directly used for the text extraction or were fed into a machine learning algorithm that learned generalized forms of the kind of expressions described by the rules. In our case, we are dealing with free-form user-generated text where there are no standard phrases or forms (e.g., "you should", "it is needed", "follow") signifying actions, that could be used for building rules based on term analysis. Neither actions in text are structured using elements, such as html tags, bullets or numbering (as in WikiHow), which could be used for creating rules based on structural analysis. In addition, the generation of rules is a complex, laborious and time consuming task. To cope with the problems above, we follow

We were recommended to an agent by friends and started looking at houses with her. In California, you can bid on more than one house at a time, so we bid on every house that we liked because the housing market is very competitive. ... Know your limits. If you're stressed out by the process, take a couple weeks off.

Figure 5.3: Textual description for the goal *buy a house*.

a two-phase semi-automatic approach that exploits only syntactic information.

[*Candidate Extraction*] Given a set of user proposals, we exploit the *syntactical structure* of the textual descriptions to identify action chunk candidates (Section 5.4.2).
[*Candidate Refinement*] For phase, we have created a dataset of phrases labeled as *actions* or *not*, and we use two alternative approaches to learn the syntactical patterns of action chunks: a data mining method that generates rules and a machine learning one that trains a classifier. We are then able to decide which candidates from the output of the first phase are indeed action chunks (Section 5.4.1).

Candidate Extraction

In considering action chunk candidates, we are looking for expressions that are concise yet meaningful and self-explanatory descriptions of actions. There are different options regarding what parts of a textual description could serve as candidate action chunks. In one extreme, one could consider terms but single terms rarely suffice as standalone, self-explanatory units of actions. For instance, "spent time searching" makes more sense than just "spent". On the other hand, sentences are standalone units but may contain more information than needed for the description of an action. Instead, our approach uses clauses and phrases as building blocks and expands them to form candidate action chunks, whose size may vary as needed in order to capture the action in a concise and meaningful way. Clauses and phrases can be produced by a syntax-tree parser such as the Stanford Penn Treebank (PTB) parser. Since we are looking for text parts that describe actions, *verbs*, and consequently, *verb phrases* comprise the core of action chunks.

Example 11 *Consider the first sentence, "*We were ... at houses with her.*", from Figure 5.3. A parse tree is an ordered, rooted tree that represents hierarchically the syntactic structure of a string. Figure 5.4 depicts the tree for this sentence produced by the Stanford Penn Treebank parser. Its root is the sentence and the leaf nodes map to tokens. Intermediate nodes correspond to phrases, such as noun phrases or verb phrases, that are split into smaller ones, each one mapping to one of its child nodes. Figure 5.5 shows the verb phrases mapping to intermediate nodes of the tree.*

We developed a recursive algorithm that takes as input a parse tree and works in two phases. In the *first phase*, the algorithm performs a *depth-first traversal*, and recursively visits all verb phrases. The output of this phase is a sequence of two kinds of chunk:

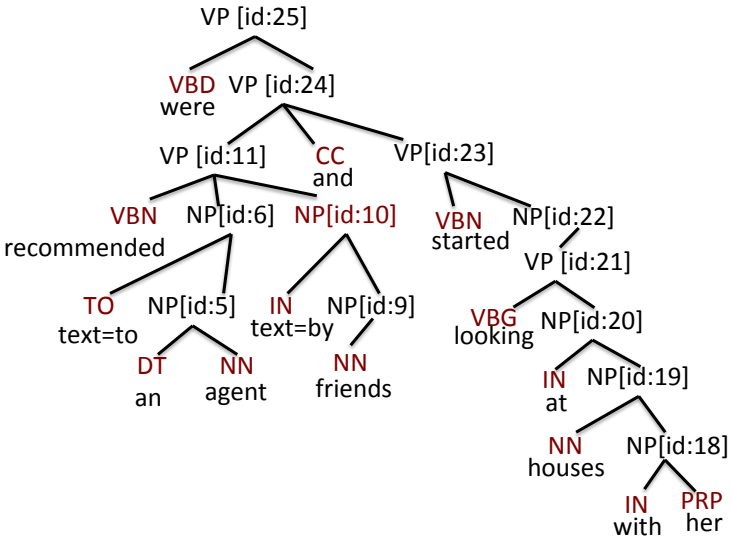


Figure 5.4: PTB parse tree.

[$id : 25$] text=were recommended to an agent by friends and started looking at houses with her
[$id : 11$] text=recommended to an agent by friends
[$id : 24$] text=recommended to an agent by friends and started looking at houses with her
[$id : 23$] text=started looking at houses with her
[$id : 21$] text=looking at houses with her

Figure 5.5: Verb phrases in the parse tree.

candidate action chunks (gist) and context chunks. The former contain a verb in one of the parser syntactic forms VB, VBD, VBG, and etc. Context phrases are related to candidate action chunks and may provide additional information. In the *second phase*, if a candidate action chunk is too short, we try to merge it with adjacent context chunks. We consider an action chunk short if it contains two or less terms. The context before or after the candidate chunks could be further exploited. However, we focus on the core of the action chunk to be able to identify in the next step, the action identification, "similar" instances. Algorithms 7 and 8 describe the extraction steps.

Example 12 *The output of the first phase of the algorithm for the first sentence in Figure 5.3 is:*

Context*:We/PRP*
  Gist*: were/VBD*
    Gist*: recommended/VBN to/TO an/DT agent/NN by/IN friends/NNS*
    Context*: and/CC*
  Gist*: started/VBN*
    Gist*: looking/VBG at/IN houses/NNS with/IN her/PRP*

Algorithm 7 *Find Candidate Action Chunks*

Input: M: A set of text chunks tagged as ROOT with at least one child that recursively contains at least one verb phrase
S: the syntax tree
Output: A set of Gist and Context Sequences

```
1  for each m in M: /* phase 1 */
2    Seqs = Seqs + Find_Gist_and_Context( m, S, false )
3  for each Seq_m in Seqs: /* phase 2 */
4    for each chunk ch in Seq_m:
5      if ch labeled as Gist and ch.Terms.length¿2:
6         VC=VC + {ch}
7      else:
8      /*check if it should be merged with
9     the next or previous chunk of the sequence */
10        newChunk = merge(previousChunk, nextChunk)
11        VC = VC + {newChunk}
```

*In this phase of the algorithm, consecutive, not self-explanatory chunks are merged. For example, "*started*" will be merged with "*looking/VBG . . . meeting/NN*". Note that some merges could be avoided without loss of information. For example, if we did not merge "*started*" above, the meaning of the action would not be lost. In other cases, merging is necessary. For instance, if "*stopped*" or "*didn't*" was written instead of "*started*", we should definitely consider this information. Therefore, we choose to perform all these merges.* The final output of the algorithm is:

Candidate Action Chunk 1: were/VBD recommended/VBN to/TO an/DT agent/NN by/IN friends/NNS

Candidate Action Chunk 2: started/VBN looking/VBG at/IN houses/NNS with/IN her/PRP

Candidate Refinement

Having generated a set of candidate action chunks, not all of them convey actions towards a goal. For example, text chunks that describe states such as "am so happy", or too general or incomplete phrases such as "Just do it", "always thought", "Yes I was naïve" should be removed.

It is easy for humans to read candidate action chunks and label them as *action* or *non-action*. Hence, we have used the output of the previous phase, i.e., a set of candidate action chunks, on a training set of textual descriptions, and we have created two manually labeled datasets (see Table 5.3 and Section5.5 for more details). The *seed dataset* is used as input for our candidate refinement to capture the syntactic patterns that distinguish action chunks from non-action ones. To do so, we follow two approaches that are both

Algorithm 8 *Gist_and_Context*: Get Sequence of Gist (chunks that may be Verb Chunks) and Context Chunks

Input: m: A text chunk that belongs in M (text chunks with at least one child that recursively contains at least one verb phrase)
S: the syntax tree
onlyLeaves: boolean
Output: A set Of Sequences of Gist And Context

```
1   if onlyLeaves=True:
2     if chunk.matches(".*.VB.*") == False:
3        /*chunk labeled as Context*/
4        Seq_m += "Context": + chunk
5     else:
6        /*chunk labeled as Gist*/
7        Seq_m += "Gist": + chunk
8     chunk=""
9   for each child ch in m.Children:
10    if ch.ContainsVP=True:
11       findGistAndContext( ch )
12    else:
13       onlyLeaves=True
14       /*get terms and POS tags*/
15       chunk+=ch.getLeavesTextAndPosTAGS
```

based on the syntax and in particular on the POS tags. In the experiments, we use the *test dataset* to test the effectiveness of this step.

With POS tags we can detect incomplete verb phrases, e.g., "that make", that are written between context and verb phrases that describe actions, and therefore should be labeled as non-action. Moreover, they can help us distinguish too generic phrases. For instance, phrases that contain only a verb and a pronoun ("'ve done them") cannot sufficiently describe an action. Some specific verb types also indicate non-actions, such as "be" and "have" that express states. We use a post-processing step for removing such phrases.

Rule-based Annotator. The first approach is to divide the seed dataset into two subsets: one with all the chunks labeled as actions, and one with all labeled as non-action, and examine the syntactic patterns in each one of them. For each action chunk, we consider the sequence of the POS tags of the words that it contains (e.g.: VB DT NN). We use a *frequent sequence mining algorithm*, namely SPADE [Zaki, 2001], to mine frequent patterns. The derived patterns can be provided to a *rule-based annotator* that will label the candidate instances according to the patterns found (or not found) in a candidate instance. Table 5.1 illustrates the results of the SPADE algorithm using as

Patterns In Instances Labeled as *Action*	"VB DT NN RB", "VB JJ NNS","VBP DT NN", "VBD DT NN", "VB DT NN", "VBN DT NN", "VB PRP$ NN" "VBD PRP$ NN", "VBD RP DT NN", "VB IN NN", "VB NN NN"
Patterns In Instances Labeled as *Non-Action*	"VB PRP RB", "VBZ CD NN", "VBP RB RB", "MD VB VBN RB", "VBP RB VB", "VBP RB VB PRP", "MD VB NN", "VBP DT JJ NN","VBD CC" "VBD RB JJ", "VB RB VB JJ", "VB IN PRP", "VBG PRP$ NN", "VB DT JJ NN", "VBZ RB JJ", "VBZ PRP$ NN"

Table 5.1: Syntactic Patterns For (Non) Action Chunks.

input the *seed dataset*.

Binary Classifier. The other approach is to automatically learn the correlations among POS tags using a machine learning technique. More specifically, we train a binary classifier to label candidate action instances as *actions* or *non-actions* using as features the POS tags of the words of the candidate instances.

Example 13 *The output of this step having as input the candidate action chunks derived from the text in Figure 5.3 is labeled as follows:*

were recommended to an agent by friends: action
started looking at houses with her: action
bid on more than one house at a time: action
is very competitive: non-action
. . .
're stressed out by the process: nonaction

5.4.2 Action Identification

With every textual description in the user proposals transformed into a set of action chunks, the next step is to assign each action chunk to the right action. Unfortunately, action chunks do not have specific structure nor attributes. They simply consist of a number of terms. Furthermore, there is no ground truth of the possible actions found in user proposals. These restrictions make several existing techniques used in entity identification not applicable.

Clustering is often used for entity identification. The objective of our task is to group action chunks into groups, where all the elements of a group describe the same action, thus, text clustering is a natural choice. Since action chunks contain a small number of terms that appear only once, more advanced weighting schemes such as tf-idf or topic-

based representations are not applicable. Moreover, a boolean model would produce very sparse vectors. Therefore, we use a set-based clustering approach, called *Set-based Centroid Clustering*, that sees each action chunk as a set of terms, and uses set-based similarity, such as Jaccard Coefficient, for similarity comparisons (Subsection 5.4.2). Set-based Centroid Clustering allows an action chunk to belong to more than one cluster. To generate the goal implementations we need to assign each action chunk to a single group, i.e., a single action.

Set-based Centroid Clustering

Set-based similarity has been used in hierarchical agglomerative clustering where a merge occurs based on the average similarity among all objects, or on the highest intra-cluster similarity (MST) [Jain and Dubes, 1988]. This requires pair-wise comparisons among all the action chunks every time a merge occurs, which is prohibitively expensive. To tackle the efficiency issue, we propose a new technique for grouping the action chunks that allows hierarchical agglomerative clustering to be applied without considering in each iteration all the action chunks. Our technique is based on a simple but important observation: action chunks that refer to the same action do not constitute separate objects as in other clustering tasks, but can be treated collectively. To do so there is a need for a representative structure (i.e., object). Typically, this representative structure is the centroid of a cluster formed by the terms of the individual action chunks the cluster contains. However, doing so leads to a well-known problem in categorical data clustering: with every merge, the number of considered terms will almost always increase and at the same time their mean value will decrease [Guha et al., 1999].

To avoid this problem we instead use an alternative structure as the centroid, which we will refer to as the *set-based centroid*. The set-based centroid of a cluster of action chunks is not formed by the intersection of the terms of the actions chunks it contains, but of the *intersection* of their *neighborhoods*, where the neighborhood of an action chunk consists of all other chunks with similarity above some threshold. Once the neighborhoods have been computed, potential actions are generated and then refined to get the final action set. These three steps are explained next.

Step A: Action Chunk Neighborhood Generation. Given the set of action chunks derived from the available user proposals, for each chunk c, we create its *neighborhood* $N(c)$ of similar chunks. For this purpose, each chunk is represented as a set of synsets derived from its terms using Wordnet[2] [Fellbaum, 1998]. We use synonyms to count for different expressions of the same action, such as "get a loan" and "borrow money". We measure chunk similarity as the Jaccard similarity of the respective sets of synsets.

[2]http://wordnet.princeton.edu

Given an action chunk c, all chunks that have a similarity higher than some threshold are considered neighbors of c.

Example 14 *Consider a set of action chunks derived from a set of user proposals: "start looking at houses around", "looked at houses", "was checking at houses, "eat healthy", "study every night". For the action chunk c: "started looking at houses with her", its stemmed terms are {start, look, house}, for which Wordnet gives three synsets. Hence, c can be represented as the following set of synsets:*
{{get-down,begin,get,start-out,start,set-about,set-out,commence}, {look, check}, {house}}. Comparing c to the synset representations of the other chunks, we determine its neighborhood $N(c)$={ { start looking at houses around}, {looked at houses}, {was checking at houses} }.

With this step, we have gone from representing each chunk as a set of terms to representing it as a set of chunks (the neighbors) reducing the dimensionality of the problem.

Step B: Potential Group Generation. In this step, we compare action chunks pairwise by comparing their neighborhoods. If the overlap of their neighborhoods, measured using the Jaccard Coefficient, is greater than a threshold, then the corresponding action chunks are merged into one by taking the union of the two neighborhoods. This union is what we call the *set-based centroid*. Note that in the current step, if the neighborhood of an action chunk (i.e., its set-based centroid) has a high overlap with the neighborhood of two other chunks (i.e., with their set-based centroid), then the same overlap will be found when comparing (the set-based centroids of) the neighborhoods of each of the two other chunks as well. Our algorithm makes sure that this overlapping redundancy is removed and outputs a set of neighborhoods of chunks. All the action chunks in a neighborhood refer to the same action, i.e., each of the resulting neighborhoods represents a different action.

Step C: Final Action Group Generation. This optimization (refinement) step is similar to the one above, only this time instead of merging neighborhoods, we merge the actual action chunks. This means that there is a series of pairwise comparisons among the set-based centroids, and if found to have a similarity higher than a threshold, the respective centroids are merged by considering the union of their action chunks. When no more merging can take place, each set-based centroid that has remained, i.e., set of action chunks, is considered as representing an action.

Conflict Resolution

Our set-based action chunk grouping algorithm may place an action chunk in more than one action clusters. This happens when a chunk is found in more than one neighborhood

Number Of Candidate Action Chunks	368874
Number of Action Chunks Labeled as Actions	165127
Number of Actions A	8378
Number of Goals G	3747
Number of Implementations I	18047

Table 5.2: Goal Implementation Extraction Results.

that are not merged with each other during the second step (either they remain alone or they are merged with other neighborhoods). This situation creates a problem since if the action chunk c appears in the description t of a goal proposal $\langle g,t\rangle$, it will not be clear which action to choose for the goal implementation, in other words, it is not clear what the result of $ref(c)$ should be.

To resolve a conflict with a chunk c, we consider all the action clusters where c belongs. In each cluster, we count the number of action chunks that are found in any goal proposal for goal g. The cluster with the largest number of such action chunks wins c. With every action chunk assigned to an action, every proposal has finally been turned into an action implementation.

5.5 Experimental Evaluation

We crawled 25K public posts from the site 43Things for the period March-September 2013. After cleaning and post deduplication, we got *18047 user proposals* for *3747 different goals*. On average, there are 7.39 user proposals per goal.

Goal Implementation Extraction. Initially 370K text subsequences (chunks) were extracted as candidate action chunks. After being pruned (candidate action refinement) by the binary classifier (using as features the POS tags), 86K remaining action chunks were transformed into 8K actions using the following three thresholds: 0.7, 0.8 and 0.8 for the corresponding three steps of the clustering (action identification procedure). After conflict resolution 18K implementations were formed. Table 5.2 summarizes the outcome of our goal implementation extraction on the dataset.

5.5.1 Comparison of Methods for Candidate Refinement

We evaluated the two alternative methods for candidate refinement presented in Section 5.4.1, namely the binary classifier and rule-based annotator, and compared them to the Conditional Random Field method. CRF has been used in the literature together with hand-crafted rules for identifying actions [Jung et al., 2010].

In order to make the comparison, we created a dataset of chunks labeled as *actions* or

non-actions. The annotations were made by two annotators and disagreement cases were re-examined until an agreement was met. We divided the annotated dataset into two sets (Table 5.3): a *seed dataset* of 486 chunks to train the classifier and mine the syntactical patterns, and a *test dataset* of 10255 chunks to evaluate the precision and recall of the task. Note that the test dataset is 21 times larger than the seed. This way we ensured that our method does not depend much on the size of the seed data.

Figure 5.6 illustrates the effectiveness of the three approaches. The *rule-based annotator* that considers only frequent action patterns gives results of very low recall (29%) for the *action* label in comparison to the *binary classifier* that gives 71.6%; while the precision values are very close: 61% and 63% respectively. Moreover, we see that by considering sequences of POS tags instead of sets of POS tags, i.e., by using CRF, the precision is not significantly improving (only 4.3%) and the recall gets worse (-7.9%). This happens because it is not common to see a chunk with a set of POS tags in some order being an action, while another chunk with the same POS tags placed in a different order being a non-action.

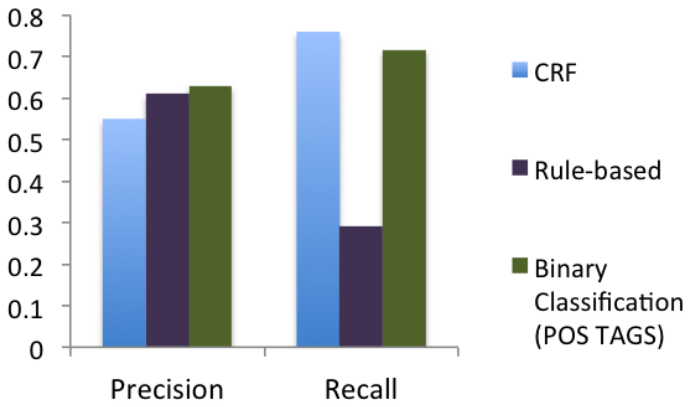


Figure 5.6: Comparison of methods for candidate refinement.

		Num of chunks annotated as	
Dataset	*Size*	*actions*	*not actions*
Seed	486	190	296
Test	10255	3123	7132

Table 5.3: Test and seed dataset for action chunk recognition.

5.5.2 Evaluation of the Final Actions

The actions derived by our methods are clusters of action chunks. To evaluate their quality, first we need to examine whether each action chunk has been placed in the right action (i.e., cluster). For this, we employ a standard clustering evaluation measure,

silhouette coefficient, which depends on the distance measure used, which in our case is the jaccard distance on the synsets of the terms of the action chunks.

We tuned all 3 thresholds used in action identification. We observed that the first one that determines the initial neighborhoods shapes the final results. By changing the other two, the coefficient remains (almost) the same. Table 5.4 illustrates the results for different threshold values. The largest coefficient value (0.947) occurs when a threshold of 0.9 is used in the first step. In practice it is also important that few action chunks remain as standalone actions. For this, we need to select a lower threshold value. A more suitable choice is 0.7 that leaves unclustered fewer chunks and at the same time gives action chunk groups with high average silhouette value (*0.868*).

Jaccard Distance Threshold (T_1)	*Average Silhouete*	*Precision*	*Recall*	*Accuracy*
0.9	0.947	0.906	0.522	0.662
0.8	0.936	0.906	0.523	0.663
0.7	0.868	0.899	0.534	0.67
0.6	0.654	0.755	0.598	0.667

Table 5.4: Evaluation of final actions (for different thresholds).

Since the silhouette coefficient depends on the jaccard distance on the synsets, we need to further examine the derived actions to see whether there are cases where the distance betwen two chunks is low but in fact they express different actions due to different context (or the other way around). For instance, the chunks *hold my breath* and *hold your breath underwater*, at first glance, seem to express the same action but when we have examined them as a part of the descriptions of the two goals: *get my eyebrow pierced* and *learn how to swim*, we have found out that they refer to two different actions (ref. Table 5.5). A true positive (TP) occurs when two action chunks expressing the same action were put in the same cluster and a true negative (TN) when they were correctly assigned to different clusters. Moreover, there are two types of error, false negative (FN) and false positive (FP). Table 5.5 gives examples for each of these cases. Since it is not possible to check the whole dataset (16K action chunks) and decide for each action chunk which other action chunks are about the same action, what we did instead was to form for each chunk (core chunk) a neighborhood consisting of its closest chunks using a relatively low Jaccard value: 0.6. Given a neighborhood, we examined whether each action chunk expresses the same action with the respective core chunk. Each pair of chunks was examined considering the goal about which they were written so as to make the judgement in the right context.

Considering 993 neighborhoods, 16373 pairs have been examined. 40% of pairs (6648 pairs) were found not to describe the same action. Given the results, we have counted

Result	Goal 1	Goal 2	Chunk 1	Chunk 2
TN	learn spanish	learn to scuba dive	take up Spanish lesson	took some lessons
TP	learn to belly dance	learn how to box	strengthen our core muscles	help strengthen my core muscles
FN	read the bible	read more books	read at night read	before bed on nights
FP	get my eyebrow pierced	learn how to swim	hold my breath	hold your breath underwater

Table 5.5: Examples of action chunks correctly (True Positives or True Negatives) or incorrectly (False Positives or False Negatives) placed in the same or different action.

the TPs, TNs, FPs, FNs and estimated precision and recall. We considered different threshold combinations as well. Table 5.4 illustrates the results. Precision values are high. For instance, for the combination of thresholds 0.7-0.8-0.8, is 0.89. Recall values are acceptable (0.53). For our problem, higher precision is more important than coverage because when two chunks take falsely the same id that also results in a false association between the respective goal implementations. There is room for further investigation in order to improve coverage. For instance, paraphrase detection could possibly further enhance the refinement step [Bhagat, 2009].

	Iterations	
T_2-T_3	HC-Baseline Cl.	Set-Based Cl.
0.5-0.5	454	14
0.6-0.5	441	10
0.5-0.6	460	8
0.6-0.6	444	7
0.7-0.5	385	4

Table 5.6: Efficiency of Set-Based Clustering.

Comparison of Set-based Centroid Clustering vs HC Clustering. We also performed an evaluation of the efficiency of our set-based cendroid mechanism for the refinement of the action chunks in comparison to plain hierarchical clustering. Since the cost of running plain hierarchical clustering on the whole action chunk set, i.e., $16K$, was prohibitive, we estimated the results for a random sample of $5K$ action chunks for both methods and we present the results for different threshold combinations in Table 5.6. We see that the number of iterations in hierarchical clustering without the use of set-based centroid is about 32 times greater.

5.5.3 Evaluation of the Goal Implementations

In addition to the action evaluation, we performed a user study in a well-known crowd-sourcing platform in order to evaluate the final outcome of our technique. With this user study, we wanted to check: (*a*) whether the action set that is extracted from the dataset is complete and correct, and (*b*) to further verify that the action chunk extraction returns meaningful chunks. We have randomly selected 40 goal descriptions (texts) and we have shown them to the participants divided in chunks (both the ones that have been used to form the actions and those that have been rejected by our candidate chunk extraction method). *Considering the respective goal*, they were asked to answer two questions: (*a*) whether *a chunk is meaningful*, and (*b*) whether *it describes an action*. The confidence of the participants for the 2 questions are 77.8% and 81.7% respectively.

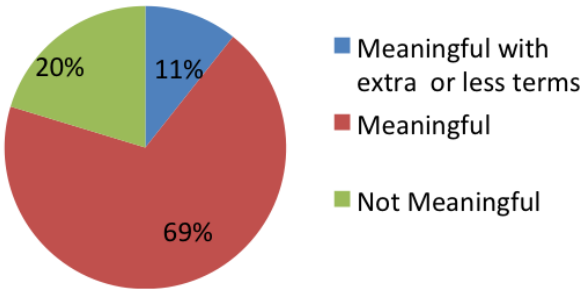


Figure 5.7: User Evaluation Of Action Chunks.

Figure 5.7 illustrates the results of the first question and Table 5.7 of the second regarding the chunks that have been used by our method to form the action sets. We observe that only 20% of the extracted chunks were characterized as non-meaningful, i.e., the derived actions consist of meaningful action chunks in most of the cases. Moreover, we observe that both precision and coverage values (48% and 64% respectively) are satisfactory given the nature of the problem but there is room for improvement. The problem of goal implementation is a new problem that opens up new challenges.

Percentage of the actions in the the goal implementations that are indeed actions	
Precision	0.64
Percentage of the actions detected by the users that are also included in the goal implementations	
Coverage	0.48

Table 5.7: User evaluation of the extracted implementations.

Chapter 6

Conclusions and Future work

Goals have already been captured and applied to improve the effectiveness of a number of different applications. In this dissertation we focus on the benefits that information systems could gain by incorporating goals in the data representation as well as in the techniques for mining, querying and retrieving items and information. Goal-aware systems can reinforce the user engagement and sense of satisfaction due to the usefulness and unexpectedness of the derived information in different domains. This dissertation deals with three different problems in the context of goal-aware systems that are aiming for retrieving items of interest from very large item collections as response to explicit user requests or in the form of recommendations.

- We proposed a novel approach for matching a reference post to related posts in a collection, i.e., finding the k most related ones. Our method identifies and exploits post segments that convey similar author intentions, i.e., they are serving the same communication goals. We presented several experiments regarding the right segmentation criteria, the effectiveness of the segmentation algorithms and the forming of the intention clusters that have proved that a rather intuitive concept, that of the authors' intentions to communicate a certain message, can be effectively captured by an automatic methodology. Moreover, due to the nature of the posts, measuring the relatedness score after having distinguished the different segments/messages that the authors intend to communicate has been proved more effective than the direct comparison of the whole posts. Specifically, our approach evaluated by real users, and in comparison with direct fulltext comparison, increased mean precision by 10%, 12% and 10.1% considering posts in a product support, a travel, and a programming forum.

- We have also introduced a family of recommendation approaches that recommend actions seeing them in respect with a number of goals that the users may fulfill

through different action sets. We have presented 3 strategies, each one incorporating goals into the scoring of actions in a different way. The action selections of the goal-based mechanisms are not affected by their domain-based similarity with the actions in the user's activity, nor by the activities of other users. However, they are affected by the benefit of the actions to be recommended to the goals in the user's goal space. The strategy *Breadth* and *Best Match* focus on more than one goals at a time. In fact the latter considers all the goals in the goal space independently from the examined action. On the other hand, the *Focus* mechanisms focus on the fulfillment of one goal at a time. Nevertheless, they all increase the average goal completeness in the user's goal space without retrieving actions that monopolize the goal implementations. Moreover, all the mechanisms create different recommendation lists for different inputs (i.e., user activities). Finally, we have mentioned several types of queries that can be answered on top of goal and action knowledge. Developing appropriate *query schemes for goal implementations* is an open challenge.

- We have also dealt with a problem strongly connected to goal-aware systems, the goal implementation extraction, i.e., the identification and modeling of the actions and the goals that they fulfill. There exists a valuable source of information about how humans accomplish the goals they set in their lives. We have described the main steps of the process, namely action chunk recognition and action identification, and we have proposed algorithms that deal with the several challenges of the problem, such as free-form text, no ground truth, to recall a few. Goal implementation extraction is a novel problem and there are a lot of challenges to handle. We will discuss some of them in Section 6.0.1.

6.0.1 Future directions in Goal-aware Systems

We believe that goal-aware approaches will and should gain more ground in the scientific community and industry. Below we discuss some of the many future research directions in the topic.

Extraction of Goal related Information: Goals, Actions and Implementations.

Building and leveraging knowledge about goals and actions is a new, complex information extraction and integration problem, where goal implementation extraction is one of the several problems. In Section 4.6 we discussed how our association-based goal model built based on a goal implementation set can be exploited to answer different queries. For a more complete answering mechanism, one may also want to look into *action normalization*, i.e., how to bring actions into a similar form. For example, the phrases "I got a loan", "we needed to get a loan" and "borrow money" have been identified as the same action.

Therefore, they could be replaced with one canonical action, such as "get a loan". Other related novel problems are: *action integration*, i.e., the integration of different action sets for the same goal, and *goal implementation visualization* that regards the presentation of alternative or even partially overlapping sets of actions for the same goal at the user interface level in a succinct way.

Goal-aware Systems.

- *Big data and query processing.* As the amount of data outgrows, the capabilities of query processing technology and the number of emerging applications, from social networks to scientific experiments, is growing fast, there is a clear need for efficient big data query processing to enable the evolution of businesses and sciences to the new era of data deluge. In this context, introducing goal-aware data and query processing methods can provide a whole new perspective into building database systems which are tailored for big data and the goals of the users accessing these data by providing features such as adaptive indexing, adaptive loading and sampling-based query processing and goal-aware query processing and optimization methods. These directions focus on reconsidering fundamental assumptions and on designing next generation database architectures for the big data era. A system, by considering for instance sets or sequences of queries that operationalize certain goals, can focus only on the part of the data that will serve the user's purpose and not every possible data that satisfy the query conditions. The query conditions, in this context, would constitute the conditions on the environment variables of the respective goals.

  The number of goals that are typically to be processed are not proportional to the volume of the data. Therefore, even goal models that are meant for a smaller number of goals, e.g., those based on consistency checking, may be easily employed. The challenging issues in this case regard the building and maintainance of the adequate structures, and the design of algorithms that will enable the answering of queries efficiently and effectively.

- *Interactive data exploration.* Interactive data exploration is an emerging form of data-intensive analytics in which users ask questions over a dataset to make sense of the data, identify interesting patterns and relationships, and bring aspects of interest into focus for further analysis. Interactive data exploration is fundamentally a multi-step, non-linear process. Data exploration requires users to possibly ask a large number of queries as they try to navigate through large data sets. Incorporating notions of goals seems like a natural step and requirement for reducing the human workload and serving better results faster. The operationalization of the goals considered by the system may contain whole queries, and/or interactions with

tuples or columns, or clicking on single fields and values. However, since users often have under-specified and shifting end-goals, goal modeling and recognition is very challenging. Hierarchical goal inference could be more appropriate in this case to capture the refinements of user goals while interacting with the system the same way they often do in dialogue systems. Systems intended for interactive data exploration can exploit knowledge about the operationalization of goals of the average user, as well as enhance goal models by information regarding the current user so as to give personalized results when needed. However, the main objective remains always to facilitate data exploration in terms of time and effort.

Bibliography

Aarts, H. and Hassin, R. R. Automatic goal inference and contagion: On pursuing goals one perceives in other people's behavior. pages 163–157, 2005.

Achtert, Elke; Kriegel, Hans-Peter; Schubert, Erich, and Zimek, Arthur. Interactive data mining with 3d-parallel-coordinate-trees. In *SIGMOD 2013, NY, USA*, pages 1009–1012, 2013.

Adomavicius, Gediminas and Tuzhilin, Alexander. Context-aware recommender systems. In *Proceedings of the 2008 ACM Conference on Recommender Systems*, RecSys '08, pages 335–336, New York, NY, USA, 2008. ACM. ISBN 978-1-60558-093-7. doi: 10.1145/1454008.1454068.

Ajzen, Icek. The theory of planned behavior. *Organizational Behavior and Human Decision Processes*, 50(2):179 – 211, 1991. ISSN 0749-5978.

Amer-Yahia, Sihem; Roy, Senjuti Basu; Chawlat, Ashish; Das, Gautam, and Yu, Cong. Group recommendation: Semantics and efficiency. *Proceedings of the VLDB Endowment*, 2(1):754–765, 2009.

Anh, Han The and Pereira, Luís nMoniz. State-of-the-art of intention recognition and its use in decision making. *AI Commun.*, 26(2):237–246, 2013.

Armentano, Marcelo and Amandi, Analía. Goal recognition with variable-order markov models. In *IJCAI*, pages 1635–1640, 2009.

Armentano, MarceloGabriel and Amandi, Anal$\sqrt{}\neq$a. Plan recognition for interface agents. *Artificial Intelligence Review*, 28(2):131–162, 2007. ISSN 0269-2821.

Atkinson, David J and Clark, Micah H. Can we engineer a human-machine social interface for trust. in trust and autonomous systems, papers from the 2013 AAAI spring symposium. volume SS-13-07 of *Technical Report*. Menlo Park: AAAI Press (2013), 2013.

Auer, S.; Bizer, C.; Kobilarov, G.; Lehmann, J.; Cyganiak, R., and Ives, Z. G. Dbpedia: A nucleus for a web of open data. In *The Semantic Web, 6th International Semantic Web Conference, 2nd Asian Semantic Web Conference, ISWC 2007 + ASWC 2007, Busan, Korea, November 11-15, 2007.*, pages 722–735, 2007.

Austin, James T and Vancouver, Jeffrey B. Goal constructs in psychology: Structure, process, and content. *Psychological bulletin*, 120(3):338, 1996.

Baeza-Yates, Ricardo; Calderan-Benavides, Liliana, and Gonzalez-Caro, Cristina. The intention behind web queries. In *String Processing and Information Retrieval*, volume 4209 of *Lecture Notes in Computer Science*, pages 98–109. Springer Berlin / Heidelberg, 2006. ISBN 978-3-540-45774-9.

Bagozzi, Richard P. and Dholakia, Utpal M. Intentional social action in virtual communities. *Journal of Interactive Marketing*, 16(2):2 – 21, 2002. ISSN 1094-9968.

Baikadi, Alok; Rowe, Jonathan P; Mott, Bradford W, and Lester, James C. Toward narrative schema-based goal recognition models for interactive narrative environments. In *Intelligent Narrative Technologies: Papers from the 2012 AIIDE Workshop AAAI Technical Report.*, 2012.

Balakrishnan, Suhrid. On-demand set-based recommendations. In *Proceedings of the Fourth ACM Conference on Recommender Systems*, RecSys '10, pages 313–316, New York, NY, USA, 2010. ACM. ISBN 978-1-60558-906-0.

Berant, Jonathan and Liang, Percy. Semantic parsing via paraphrasing. In *ACL (1)*, pages 1415–1425, 2014.

Bhagat, Rahul. *Learning Paraphrases from Text.* PhD thesis, Los Angeles, CA, USA, 2009. AAI3368694.

Blaylock, Nate and Allen, James. Corpus-based, statistical goal recognition. In *IJCAI 2003*, pages 1303–1308. Morgan Kaufmann Publishers Inc., 2003.

Blaylock, Nate and Allen, James. Recognizing instantiated goals using statistical methods. In *G. Kaminka (Ed.), Workshop on Modeling Others from Observations (MOO2005)*, pages 79–86, 2005.

Blei, D. M. Probabilistic topic models. *Commun. ACM*, 55(4):77–84, April 2012.

Blei, D. M.; Ng, A. Y., and Jordan, M. I. Latent dirichlet allocation. *J. Mach. Learn. Res.*, 3:993–1022, 2003.

Bossart, JL and Prowell, D Pashley. Genetic estimates of population structure and gene flow: limitations, lessons and new directions. *Trends in Ecol. & Evol.*, 13(5):202–206, 1998.

Broder, Andrei. A taxonomy of web search. *SIGIR Forum*, 36(2):3–10, 2002. ISSN 0163-5840.

Carberry, Sandra. Tracking user goals in an information-seeking environment. In *AAAI*, pages 59–63, 1983.

Carberry, Sandra. Techniques for plan recognition. *User Modeling and User-Adapted Interaction*, 11(1-2):31–48, 2001a. ISSN 0924-1868.

Carberry, Sandra. Techniques for plan recognition. *User Modeling and User-Adapted Interaction*, 11(1-2):31–48, 2001b.

Carpineto, Claudio; Mizzaro, Stefano; Romano, Giovanni, and Snidero, Matteo. Mobile information retrieval with search results clustering: Prototypes and evaluations. *Journal of the American Society for Information Science and Technology*, 60(5):877–895, 2009. ISSN 1532-2890.

Castellanos, Malú; Hsu, Meichun; Dayal, Umeshwar; Ghosh, Riddhiman; Dekhil, Mohamed; Limon, Carlos Ceja; Puchi, Marcial, and Ruiz, Perla. Intention insider: discovering people's intentions in the social channel. In *EDBT*, pages 614–617, 2012.

Chang, Yao-Sheng; He, Kuan-Yu; Yu, Scott, and Lu, Wen-Hsiang. Identifying user goals from web search results. In *Web Intelligence.*, pages 1038–1041, 2006.

Charniak, Eugene and Goldman, Robert P. Plan recognition in stories and in life. *CoRR*, abs/1304.1497, 2013.

Chelmis, C. and Prasanna, V.K. Predicting communication intention in social networks. In *Int. Confernece on Social Computing (SocialCom)*, pages 184–194, 2012.

Chen, M.; Jin, X., and Shen, D. Short text classification improved by learning multi-granularity topics. In *IJCAI*, pages 1776–1781, 2011. ISBN 978-1-57735-515-1. doi: 10.5591/978-1-57735-516-8/IJCAI11-298.

Cheung, Christy M.K. and Lee, Matthew K.O. A theoretical model of intentional social action in online social networks. *Decision Support Systems*, 49(1):24 – 30, 2010. ISSN 0167-9236.

Chow, Wing S. and Chan, Lai Sheung. Social network, social trust and shared goals in organizational knowledge sharing. *Information & Management*, 45(7):458 – 465, 2008. ISSN 0378-7206.

Chulef, Ada S.; Read, Stephen J., and Walsh, David A. A hierarchical taxonomy of human goals. *Motivation and Emotion*, 25(3):191–232, 2001. ISSN 1573-6644. doi: 10.1023/A:1012225223418.

Crook, Paul A. and Lemon, Oliver. Representing uncertainty about complex user goals in statistical dialogue systems. In *SIGDIAL Conf.*, pages 209–212, 2010.

Dalpiaz, F.; Souza, V. E. Silva, and Mylopoulos, J. The many faces of operationalization in goal-oriented requirements engineering. In *APCCM*. 2014.

De Choudhury, M.; Sundaram, H.; John, A., and Seligmann, D.D. Contextual prediction of communication flow in social networks. In *Web Intelligence 2007*, pages 57–65, 2007.

Della Pietra, S.; Della Pietra, V., and Lafferty, J. Inducing features of random fields. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 19(4):380–393, 1997. ISSN 0162-8828.

Deshpande, Mukund and Karypis, George. Item-based top-n recommendation algorithms. *ACM Trans. Inf. Syst.*, 22(1):143–177, January 2004. ISSN 1046-8188.

Doucet, Arnaud; Freitas, Nando de; Murphy, Kevin P., and Russell, Stuart J. Rao-blackwellised particle filtering for dynamic bayesian networks. In *Uncertainty in Artificial Intelligence (UAI)*, pages 176–183. Morgan Kaufmann Publishers Inc., 2000. ISBN 1-55860-709-9.

Dragunov, Anton N.; Dietterich, Thomas G.; Johnsrude, Kevin; Mclaughlin, Matthew; Li, Lida, and Herlocker, Jonathan L. Tasktracer: a desktop environment to support multi-tasking knowledge workers. In *Proceedings of the 10th international conference on Intelligent user interfaces*, pages 75–82. ACM Press, 2005.

Ester, M.; Kriegel, H.; Sander, J., and Xu, X. A density-based algorithm for discovering clusters in large spatial databases with noise. In *PODS*, pages 226–231, 1996.

Fagin, R. Combining fuzzy information from multiple systems. In *PODS*, pages 216–226, 1996.

Fellbaum, Christiane, editor. *WordNet: an electronic lexical database*. MIT Press, 1998.

Fikes, Richard E. and Nilsson, Nils J. Strips: A new approach to the application of theorem proving to problem solving. In *IJCAI*, pages 608–620. Morgan Kaufmann Publishers Inc., 1971.

Fishbein, Martin and Ajzen, Icek. *Belief, Attitude, Intention, and Behavior: An Introduction to Theory and Research.* Addison-Wesley, 1975.

Forbes, Peter and Zhu, Mu. Content-boosted matrix factorization for recommender systems: Experiments with recipe recommendation. In *Proceedings of the Fifth ACM Conference on Recommender Systems*, RecSys '11, pages 261–264, New York, NY, USA, 2011. ACM. ISBN 978-1-4503-0683-6. doi: 10.1145/2043932.2043979.

Fouss, F.; Pirotte, A.; m. Renders, J., and Saerens, M. Random-walk computation of similarities between nodes of a graph with application to collaborative recommendation. *IEEE Transactions on Knowledge and Data Engineering*, 19(3):355–369, March 2007. ISSN 1041-4347. doi: 10.1109/TKDE.2007.46.

Fujii, Atsushi. Modeling anchor text and classifying queries to enhance web document retrieval. In *WWW*, pages 337–346. ACM, 2008. ISBN 978-1-60558-085-2.

Ganesan, Kavita and Zhai, ChengXiang. Opinion-based entity ranking. *Information Retrieval*, 2011. doi: 10.1007/s10791-011-9174-8.

Geib, C. W. and Goldman, R.P. Plan recognition in intrusion detection systems. In *DARPA Information Survivability Conf. and Exposition (DISCEX)*, 2001.

Gold, Kevin. Training goal recognition online from low-level inputs in an action-adventure game. In *AIIDE*, 2010.

Govindaraju, Vidhya and Ramanathan, Krishnan. Similar document search and recommendation. *Journal of Emerging Technologies in Web Intelligence*, 4(1):84–93, 2012.

Guha, Sudipto; Rastogi, Rajeev, and Shim, Kyuseok. Rock: A robust clustering algorithm for categorical attributes. In *Proceedings of the 15th International Conference on Data Engineering*, ICDE '99, pages 512–, Washington, DC, USA, 1999. IEEE Computer Society. ISBN 0-7695-0071-4.

Ha, Eun Y.; Rowe, Jonathan P.; Mott, Bradford W., and Lester, James C. Goal recognition with markov logic networks for player-adaptive games. In *AAAI*, 2012.

Hagen, Matthias; Potthast, Martin; Stein, Benno, and Bräutigam, Christof. Query segmentation revisited. In *In*, WWW '11, pages 97–106, New York, NY, USA, 2011. ACM. ISBN 978-1-4503-0632-4. doi: 10.1145/1963405.1963423.

Han, The Anh and Pereira, Luís Moniz. Collective intention recognition and elder care. In *Proactive Assistant Agents, Papers from the 2010 AAAI Fall Symposium, Arlington, Virginia, USA, November 11-13, 2010*, 2010.

Hassan, Ahmed; Jones, Rosie, and Klinkner, Kristina Lisa. Beyond dcg: user behavior as a predictor of a successful search. In *WSDM*, pages 221–230. ACM, 2010. ISBN 978-1-60558-889-6.

He, Yulan. Goal detection from natural language queries. In *on Applications of natural language to information systems (NLDB)*, pages 157–168. Springer-Verlag, 2010. ISBN 3-642-13880-2, 978-3-642-13880-5.

Hearst, M. Texttiling: Segmenting text into multi-paragraph subtopic passages. *Computational Ling.*, 23:33–64, '97.

Heckerman, David. A tutorial on learning with bayesian networks. Technical report, Learning in Graphical Models, 1996.

Herrera, Mauro Rojas; de Moura, Edleno Silva; Cristo, Marco; Silva, Thomaz Philippe C., and da Silva, Altigran Soares. Exploring features for the automatic identification of user goals in web search. *Inf. Process. Manage.*, 46(2):131–142, 2010.

Hoelzl, Gerold; Kurz, Marc, and Ferscha, Alois. Goal oriented opportunistic recognition of high-level composed activities using dynamically configured hidden markov models. *Procedia Computer Science*, 10(0):308 – 315, 2012. ISSN 1877-0509.

Hong, Jun. Plan recognition through goal graph analysis. In *ECAI*, pages 496–500, 2000.

Hong, L. and Davison, B. A classification-based approach to question answering in discusion boards. In *ACM SIGIR '09*, pages 171 – 177, 2009.

Horvitz, E.; Breese, J.; Heckerman, D.; Hovel, D., and Rommelse, K. The lumiere project: Bayesian user modeling for inferring the goals and needs of software users. In *Proc. of Conf. on Uncertainty in Artificial Intelligence*, pages 256–265, 1998.

Hsu, Chin-Lung and Lin, Judy Chuan-Chuan. Acceptance of blog usage: The roles of technology acceptance, social influence and knowledge sharing motivation. *Information & Management*, 45(1):65 – 74, 2008. ISSN 0378-7206.

Hu, X.; Sun, N.; Zhang, C., and Chua, T. Exploiting internal and external semantics for clustering short texts using world knowledge. In *CIKM*, pages 919–928, 2009. ISBN 978-1-60558-512-3.

Huber, Marcus J. and Simpson, Richard. Plan recognition to aid the visually impaired. In *Proc. of the 9th international conference on User modeling*, UM'03, pages 138–142. Springer-Verlag, 2003. ISBN 3-540-40381-7.

Hulpus, I.; Hayes, C.; Karnstedt, M., and Greene, D. Unsupervised graph-based topic labelling using dbpedia. In *WSDM*, pages 465–474, 2013.

Jain, Anil K. and Dubes, Richard C. *Algorithms for Clustering Data*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1988. ISBN 0-13-022278-X.

Jansen, Bernard J.; Booth, Danielle L., and Spink, Amanda. Determining the informational, navigational, and transactional intent of web queries. *Inf. Proc. and Mangmt.*, 44(3):1251 – 1266, 2008. ISSN 0306-4573. doi: 10.1016/j.ipm.2007.07.015.

Jenders, Maximilian; Krestel, Ralf, and Naumann, Felix. Which answer is best?: Predicting accepted answers in mooc forums. In *Proceedings of the 25th International Conference Companion on World Wide Web*, pages 679–684. International World Wide Web Conferences Steering Committee, 2016.

Jeon, Jiwoon; Croft, W. Bruce, and Lee, Joon Ho. Finding semantically similar questions based on their answers. In *Proceedings of the 28th ACM SIGIR Conference*, SIGIR '05, pages 617–618, New York, NY, USA, 2005. ACM. ISBN 1-59593-034-5.

Jones, K.S.; Van Rijsbergen, C.J.; Research, British Library., and Department, Development. *Report on the Need for and Provision of an Ideal Information Retrieval Test Collection*. British Library Research and Development reports. 1975.

Jung, Yuchul; Ryu, Jihee; Kim, Kyung-min, and Myaeng, Sung-Hyon. Automatic construction of a large-scale situation ontology by mining how-to instructions from the web. *Web Semant.*, 8(2-3):110–124, July 2010. ISSN 1570-8268. doi: 10.1016/j.websem.2010.04.006.

Kabanza, Froduald; Bellefeuille, Philipe; Bisson, Francis; Benaskeur, Abder Rezak, and Irandoust, Hengameh. Opponent behaviour recognition for real-time strategy games. In *Plan, Activity, and Intent Recognition*, volume WS-10-05 of *AAAI Workshops*. AAAI, 2010.

Kang, In-Ho and Kim, GilChang. Query type classification for web document retrieval. In *Proc. of ACM SIGIR*, pages 64–71. ACM, 2003. ISBN 1-58113-646-3.

Kautz, Henry A. Reasoning about plans. chapter A formal theory of plan recognition and its implementation, pages 69–124. Morgan Kaufmann Publishers Inc., 1991. ISBN 1-55860-137-6.

Kazantseva, A. and Szpakowicz, S. Topical segmentation: A study of human performance and a new measure of quality. In *HLT*, pages 211–220, 2012.

Kekalainen, Jaana. Binary and graded relevance in ir. *Inf. Processing & Management*, 41(5):1019 – 1033, 2005. ISSN 0306-4573.

Keren, S.; Gal, A., and Karpas, E. Goal recognition design for non-optimal agents. In *AAAI*, pages 3298–3304, 2015.

Keren, Sarah; Gal, Avigdor, and Karpas, Erez. Privacy preserving plans in partially observable environments. In *IJCAI*, 2016a.

Keren, Sarah; Gal, Avigdor, and Karpas, Erez. Goal recognition design with non-observable actions. In *AAAI 16, February 12-17, 2016, Phoenix, Arizona, USA.*, pages 3152–3158, 2016b.

Kofler, Christoph; Larson, Martha, and Hanjalic, Alan. User intent in multimedia search: A survey of the state of the art and future challenges. *ACM Comput. Surv.*, 49(2):36:1–36:37, August 2016. ISSN 0360-0300. doi: 10.1145/2954930. URL `http://doi.acm.org/10.1145/2954930`.

Koutrika, Georgia; Bercovitz, Benjamin, and Garcia-Molina, Hector. Flexrecs: Expressing and combining flexible recommendations. In *Proceedings of the 2009 ACM SIGMOD International Conference on Management of Data*, SIGMOD '09, pages 745–758, New York, NY, USA, 2009. ACM. ISBN 978-1-60558-551-2. doi: 10.1145/1559845.1559923.

Koutrika, Georgia; Papadimitriou, Dimitra, and Simske, Steven J. Matching of an input document to documents in a document collection, December 6 2013. US Patent App. 15/100,918.

Lee, Uichin; Liu, Zhenyu, and Cho, Junghoo. Automatic identification of user goals in web search. In *WWW*, pages 391–400. ACM, 2005. ISBN 1-59593-046-9.

Lesh., Neal. *Scalable and Adaptive Goal Recognition.* PhD thesis, University of Washingto, 1998.

Lesh, Neal; Rich, Charles, and Sidner, Candace L. Using plan recognition in human-computer collaboration. In *Proc. of Int. Conf. on User modeling*, pages 23–32, Secaucus, NJ, USA, 1999. Springer-Verlag New York, Inc. ISBN 3-211-83151-7.

Li, Jiye; Tang, Bin, and Cercone, Nick. Applying association rules for interesting recommendations using rule templates. In *Advances in Knowledge Discovery and Data Mining*, pages 166–170. Springer, 2004.

Li, Yunyao; Krishnamurthy, Rajasekar; Vaithyanathan, Shivakumar, and Jagadish, H. V. Getting work done on the web: Supporting transactional queries. In *SIGIR 2006*, 2006.

Lieberman, Henry. User interface goals, ai opportunities. *AI Magazine*, 30(4):16–22, 2009.

Louvigne, S.; Rubens, N.; Anma, F., and Okamoto, T. Utilizing social media for goal setting based on observational learning. In *ICALT*, pages 736–737, 2012. doi: 10.1109/ICALT.2012.115.

Lowd, Daniel and Davis, Jesse. Learning markov network structure with decision trees. In *Proc. of Int. Conf. on Data Mining (ICDM)*, pages 334–343, 2010.

Lu, Yumao; Peng, Fuchun; Li, Xin, and Ahmed, Nawaaz. Coupling feature selection and machine learning methods for navigational query identification. In *Int. conference on Information and knowledge management (CIKM)*, pages 682–689. ACM, 2006. ISBN 1-59593-433-2.

Maragoudakis, M.; Thanopoulos, A., and Fakotakis, N. Meteobayes: Effective plan recognition in a weather dialogue system. *Intelligent Systems, IEEE*, 22(1):67–77, 2007. ISSN 1541-1672.

Meehan, James R. Tale-spin. In Schank, R., editor, *Inside Computer Understanding*, pages 197–225. Hillsdale, NJ: Lawrence Erlbaum, 1981.

Middleton, Stuart E; De Roure, David, and Shadbolt, Nigel R. Ontology-based recommender systems. In *Handbook on ontologies*, pages 779–796. Springer, 2009.

Misra, H.; Yvon, F.; Jose, J. M., and Cappe, O. Text segmentation via topic modeling: an analytical study. In *CIKM*, pages 1553–1556, 2009.

Mott, Bradford; Lee, Sunyoung, and Lester, James. Probabilistic goal recognition in interactive narrative environments. In *Artificial intelligence*, pages 187–192. AAAI Press, 2006. ISBN 978-1-57735-281-5.

Murphy, Kevin P.; Weiss, Yair, and Jordan, Michael I. Loopy belief propagation for approximate inference: An empirical study. In *Uncertainty in Artificial Intelligence (UAI)*, pages 467–475. Morgan Kaufmann Publishers Inc., 1999. ISBN 1-55860-614-9.

Mylopoulos, John; Chung, Lawrence, and Yu, Eric. From object-oriented to goal-oriented requirements analysis. *Commun. ACM*, 42(1):31–37, 1999. ISSN 0001-0782.

Newell, Allen. The knowledge level. *Artif. Intell.*, 18(1):87–127, 1982.

Ntoutsi, Eirini; Stefanidis, Kostas; Nørvåg, Kjetil, and Kriegel, Hans-Peter. Fast group recommendations by applying user clustering. In *International Conference on Conceptual Modeling*, pages 126–140. Springer, 2012.

Papadimitriou, D. Goal-aware data management for retrieval and recommendations. In *2016 IEEE 32nd ICDE Workshops*, pages 216–220, May 2016. doi: 10.1109/ICDEW.2016.7495651.

Papadimitriou, D.; Velegrakis, Y.; Koutrika, G., and Mylopoulos, J. Goals in social media, information retrieval and intelligent agents. In *2015 IEEE 31st International Conference on Data Engineering*, pages 1538–1540, April 2015.

Papadimitriou, D.; Koutrika, G.; Mylopoulos, J., and Velegrakis, Y. The goal behind the action: Towards goal-aware systems and applications. *ACM Trans. Database Syst.*, 41(4):1–43, 2016.

Pareti, Paolo; Klein, Ewan, and Barker, Adam. A semantic web of know-how: Linked data for community-centric tasks. In *Proceedings of the 23rd International Conference on World Wide Web*, WWW '14 Companion, pages 1011–1016, New York, NY, USA, 2014. ACM. ISBN 978-1-4503-2745-9. doi: 10.1145/2567948.2578846.

Parra-Lopez, Eduardo; Bulchand-Gidumal, Jacques; Gutierrez-Tano, Desiderio, and Diaz-Armas, Ricardo. Intentions to use social media in organizing and taking vacation trips. *Computers in Human Behavior*, 27(2):640 – 654, 2011. ISSN 0747-5632.

Passonneau, Rebecca J. and Litman, Diane J. Intention- based segmentation: Human reliability and correlation with linguistic cues. In *ACL*, pages 148–155, 1993. doi: 10.3115/981574.981594.

Patterson, Donald J.; Liao, Lin; Fox, Dieter, and Kautz, Henry. Inferring high-level behavior from low-level sensors. pages 73–89, 2003.

Perkowitz, Mike; Philipose, Matthai; Fishkin, Kenneth, and Patterson, Donald J. Mining models of human activities from the web. In *Proceedings of the 13th International Conference on World Wide Web*, WWW '04, pages 573–582, New York, NY, USA, 2004. ACM. ISBN 1-58113-844-X.

Perugini, Marco and Bagozzi, Richard P. The role of desires and anticipated emotions in goal-directed behaviours: Broadening and deepening the theory of planned behaviour. *British Journal of Social Psychology*, 40(1):79–98, 2001.

Ramırez, Miquel and Geffner, Hector. Plan recognition as planning. In *Proc. of the 21st international joint conference on Artifical intelligence. Morgan Kaufmann Publishers Inc*, pages 1778–1783, 2009.

Ramírez, Miquel and Geffner, Hector. Goal recognition over pomdps: Inferring the intention of a pomdp agent. In *Proceedings of the Twenty-Second International Joint Conference on Artificial Intelligence - Volume Volume Three*, IJCAI'11, pages 2009–2014. AAAI Press, 2011. ISBN 978-1-57735-515-1.

Raux, Antoine and Ma, Yi. Efficient probabilistic tracking of user goal and dialog history for spoken dialog systems. In *INTERSPEECH*, pages 801–804, 2011.

Rendle, Steffen; Freudenthaler, Christoph, and Schmidt-Thieme, Lars. Factorizing personalized markov chains for next-basket recommendation. In *Proceedings of the 19th International Conference on World Wide Web*, WWW '10, pages 811–820, New York, NY, USA, 2010. ACM. ISBN 978-1-60558-799-8.

Riedel, Sebastian. Improving the accuracy and efficiency of map inference for markov logic. *Empirical methods in natural language processing*, 2012.

Riedl, Mark Owen. *Narrative generation: balancing plot and character.* PhD thesis, North Carolina State University, 2004.

Ritter, Alan; Clark, Sam; Mausam, , and Etzioni, Oren. Named entity recognition in tweets: An experimental study. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing, EMNLP 2011*, pages 1524–1534, 2011.

Robertson, S.; Walker, S., and Hancock-Beaulieu, M. Okapi at TREC-7: Automatic ad hoc, filtering, VLC and interactive track. *TREC '98*, pages 199–210, 1998.

Ron, Dana; Singer, Yoram, and Tishby, Naftali. Learning probabilistic automata with variable memory length. In *Proc. of the 7thAnnual ACM Conf. on Computational Learning Theory*, pages 35–46. ACM Press, 1994.

Rose, Daniel E. and Levinson, Danny. Understanding user goals in web search. In *WWW*, pages 13–19. ACM, 2004. ISBN 1-58113-844-X.

Roy, Patrice; Bouchard, Bruno; Bouzouane, Abdenour, and Giroux, Sylvain. A hybrid plan recognition model for alzheimer's patients: interleaved-erroneous dilemma. In *International Conference on Intelligent Agent Technology 2007*, pages 131–137. IEEE Computer Society, 2007.

Russell, Stuart J. and Norvig, Peter. *Artificial Intelligence: A Modern Approach.* Pearson Education, 2003. ISBN 0137903952.

Ryu, Jihee; Jung, Yuchul; Kim, Kyung-min, and Myaeng, Sung H. Automatic Extraction of Human Activity Knowledge from Method-Describing Web Articles. *Proceedings of the 1st Workshop on Automated Knowledge Base Construction*, 2010.

Sadikov, E.; Madhavan, J.; Wang, L., and Halevy, A. Clustering query refinements by user intent. WWW '10, pages 841–850, New York, NY, USA, 2010. ACM. ISBN 978-1-60558-799-8. doi: 10.1145/1772690.1772776.

Sadri, Fariba. Intention recognition in agents for ambient intelligence: Logic-based approaches. In *Agents and Ambient Intelligence*, pages 197–236. 2012.

Sadri, Fariba. *Handbook of Research on Ambient Intelligence and Smart Environments: Trends and Perspectives*, chapter Logic-Based Approaches to Intention Recognition, pages 346–375. N. Chong, F. Mastrogiovanni, 2014.

Salton, G.; Singhal, A.; Buckley, C., and Mitra, M. Automatic text decomposition using text segments and text themes. In *ACM Hypertext*, pages 53–65, 1996.

Sandvig, J. J.; Mobasher, Bamshad, and Burke, Robin. Robustness of collaborative recommendation based on association rule mining. In *Proceedings of the 2007 ACM Conference on Recommender Systems*, RecSys '07, pages 105–112, New York, NY, USA, 2007. ACM. ISBN 978-1-59593-730–8. doi: 10.1145/1297231.1297249.

Sarwar, Badrul; Karypis, George; Konstan, Joseph, and Riedl, John. Item-based collaborative filtering recommendation algorithms. In *Proceedings of the 10th International Conference on World Wide Web*, WWW '01, pages 285–295, New York, NY, USA, 2001. ACM. ISBN 1-58113-348-0.

Schank, Roger C. *Tell Me a Story: Narrative and Intelligence*. Northwestern University Press, 1995. ISBN 0810113139.

Schmidt, Charles F.; Sridharan, N. S., and Goodson, John L. The plan recognition problem: An intersection of psychology and artificial intelligence. *Artif. Intell.*, 11(1-2):45–83, 1978.

Sen, Amartya K; Last, AGM, and Quirk, Randolph. Prediction and economic theory [and discussion]. *Proc. of the Royal Society of London. A. Mathematical and Physical Sciences*, 407(1832):3–23, 1986.

Shtok, A.; Dror, G., and Maarek, Y. Learning from the past: Answering new questions with past answers. In *WWW*, pages 759–768, 2012.

Singer, Yoram and Warmuth, Manfred K. Training algorithms for hidden markov models using entropy based distance functions. In *Advances in Neural Information Processing Systems 9*, pages 641–647. MIT Press, 1996.

Singh, A.; Deepak, P., and Raghu, D. Retrieving similar discussion forum threads: a structure based approach. In *ACM SIGIR*, pages 135 – 144, 2012.

Smith, Dustin A. and Lieberman, Henry. The why ui: Using goal networks to improve user interfaces. In *Proceedings of the 15th International Conference on Intelligent User Interfaces*, IUI '10, pages 377–380, New York, NY, USA, 2010a. ACM. ISBN 978-1-60558-515-4. doi: 10.1145/1719970.1720035.

Smith, Dustin A. and Lieberman, Henry. The why ui: using goal networks to improve user interfaces. In *IUI (Intelligent User Interfaces) '10*, pages 377–380. ACM, 2010b. ISBN 978-1-60558-515-4.

Stefanidis, Kostas; Shabib, Nafiseh; Nørvåg, Kjetil, and Krogstie, John. Contextual recommendations for groups. In *International Conference on Conceptual Modeling*, pages 89–97. Springer, 2012.

Strohmaier, Markus. Purpose tagging: Capturing user intent to assist goal-oriented social search. In *Proceedings of the 2008 ACM Workshop on Search in Social Media*, SSM '08, pages 35–42, New York, NY, USA, 2008. ACM. ISBN 978-1-60558-258-0.

Strohmaier, Markus; Kröll, Mark, and Körner, Christian. Automatically annotating textual resources with human intentions. In *Proceedings of the Hypertext 2009*, pages 355–356, 2009.

Suchanek, Fabian M.; Kasneci, Gjergji, and Weikum, Gerhard. YAGO: A large ontology from wikipedia and wordnet. *J. Web Sem.*, 6(3):203–217, 2008.

Sun, Jigui and Yin, Minghao. Recognizing the agent's goals incrementally: planning graph as a basis. *Frontiers of Computer Science in China*, 1(1):26–36, 2007. ISSN 1673-7350.

Teng, Chun-Yuen; Lin, Yu-Ru, and Adamic, Lada A. Recipe recommendation using ingredient networks. In *Proceedings of the 4th Annual ACM Web Science Conference*, WebSci '12, pages 298–307, New York, NY, USA, 2012. ACM. ISBN 978-1-4503-1228-8. doi: 10.1145/2380718.2380757.

Thompson, James D and McEwen, William J. Organizational goals and environment: Goal-setting as an interaction process. *American Sociological Review*, 23(1):23–31, 1958.

Wainwright, Martin J. and Jordan, Michael I. Graphical models, exponential families, and variational inference. *Found. Trends Mach. Learn.*, 1(1-2):1–305, January 2008. ISSN 1935-8237.

Wang, K.; Ming, Z., and Chua, T. A syntactic tree matching approach to find similar questions in community QA services. In *ACM SIGIR*, pages 187 – 194, 2009.

Wang, K.; Ming, Z.; Hu, X., and Chua, T. Segmentation of multi-sentence questions: towards effective question retrieval in cQA services. In *ACM SIGIR*, 2010.

Weber, Ingmar; Ukkonen, Antti, and Gionis, Aris. Answers, not links: Extracting tips from yahoo! answers to address how-to web queries. In *Proceedings of the Fifth ACM International Conference on Web Search and Data Mining*, WSDM '12, pages 613–622, New York, NY, USA, 2012. ACM. ISBN 978-1-4503-0747-5.

Wen, H.; Zhongyuan, W.; Haixun, W.; Kai, Z., and Xiaofang, Z. Short text understanding through lexical-semantic analysis. In *IEEE ICDE*, 2015.

Weng, Jianshu and Lee, Bu-Sung. Event detection in twitter. In *Proceedings of the Fifth International Conference on Weblogs and Social Media, Barcelona, Catalonia, Spain, July 17-21, 2011*, 2011.

Weng, Linkai; Li, Zhiwei; Cai, Rui; Zhang, Yaoxue; Zhou, Yuezhi; Yang, Laurence T., and Zhang, Lei. Query by document via a decomposition-based two-level retrieval approach. In *In*. Association for Computing Machinery, Inc., July 2011.

Wilcox, L.D. and Bush, M. Training and search algorithms for an interactive wordspotting system. In *Acoustics, Speech, and Signal Processing, IEEE Int. Conf. on*, pages 97–100 vol.2, 1992.

Wu, Fei and Weld, Daniel S. Autonomously semantifying wikipedia. In *Proceedings of the Sixteenth ACM Conference on Information and Knowledge Management, CIKM 2007, Lisbon, Portugal, November 6-10, 2007*, pages 41–50, 2007.

Yin, Jie; Yang, Qiang, and Pan, Jeffrey Junfeng. Sensor-based abnormal human-activity detection. *IEEE Transactions on Knowledge and Data Engineering (TKDE)*, 20(8):17–31, 2007.

Yin, Jie; Yang, Qiang; Shen, Dou, and Li, Ze-Nian. Activity recognition via user-trace segmentation. *ACM Trans. Sen. Netw.*, 4(4):19:1–19:34, September 2008. ISSN 1550-4859.

Zaki, Mohammed J. Spade: An efficient algorithm for mining frequent sequences. *Mach. Learn.*, 42(1-2):31–60, January 2001. ISSN 0885-6125.

Zhe, Shandian; Xia, Tian, and Cheng, Xueqi. Modeling users' information goal transitions and satisfaction judgment: Understanding the full search process. In *Web Intelligence*, pages 431–434, 2010.

Zhou, Tom Chao; Lin, Chin-Yew; King, Irwin; Lyu, Michael R; Song, Young-In, and Cao, Yunbo. Learning to suggest questions in online forums. In *AAAI*, 2011.